ESD-TR-65-575

RESEARCH INTO THE MANAGEMENT OF COMPUTER PROGRAMMING: A TRANSITIONAL ANALYSIS OF COST ESTIMATION TECHNIQUES

G. F. Weinwurm
H. J. Zagorski

November 1965

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

(Prepared under Contract No. AF 19(628)-5166, by Systems Development Corporation, 2500 Colorado Ave., Santa Monica, California)

ESD-TR-65-575

RESEARCH INTO THE MANAGEMENT OF COMPUTER PROGRAMMING: A TRANSITIONAL ANALYSIS OF COST ESTIMATION TECHNIQUES

G. F. Weinwurm
H. J. Zagorski

November 1965

DIRECTORATE OF COMPUTERS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

(Prepared under Contract No. AF 19(628)-5166, by Systems Development Corporation, 2500 Colorado Ave., Santa Monica, California)

FOREWORD

This report was prepared by the Systems Development Corporation (SDC), 2500 Colorado Ave., Santa Monica, California under contract AF19(628)-5166, for the Director of Computers, Electronic Systems Division (ESD), as part of a continuing research effort directed toward the development of guidelines, standards, and techniques for improved management in the field of computer programming. This research is a continuation of the work that has been sponsored by ESD since March 1964. SDC report number TM-2712/000/00 applies. The Air Force Program Monitor is Capt John W. O'Grady, ESRCC.

This report is intended as a companion document to the forthcoming "Equations for Estimating the Cost of Computer Program Production". The follow-on document (due 4 April 1966) is intended for the user whereas the present work stresses the details of the project analysis work to date.

The research reported herein was conducted at SDC as a part of the Programming Management Project. Victor LaBolle has been the leader and main motivating force behind this work since its inception in late 1962, and particular recognition is due his contributions. The initial identification of factors that affect the cost of computer programming was done by L. Farr and B. Nanus. L. Farr and H. J. Zagorski were responsible for the first statistical analysis of historical cost data in this project. N. E. Willmorth also contributed to these studies. The results of this early work are referenced in the Bibliography.

Victor LaBolle contributed material to Sections III, VIII and XX. Two relatively new Project members prepared other portions of this document: the section dealing with the sensitivity of costs to changes in the indices was prepared by Edward A. Nelson; Thomas Fleishman bore the main responsibility for the Appendices.

Drs. Lee Christie, Harry Harman, Robert McCormack, and John Walsh consulted with us on special problems involving statistical analysis.

The following were among our reviewers, who bear no responsibility for any deficiencies of this document, but whose contributions were appreciated: Dr. Lee Christie, Dr. Harry H. Harman, Dr. Robert McCormack, Jules Schwartz, Clarence Starkey and Dr. John Walsh.

Last, but certainly not least, were the contributions of our editor, Ann Walker, and typists, Carol Castillo and Diane Mitchell.

This technical report has been reviewed and is approved.

CHARLES A. LAUSTRUP, Colonel, USAF
Director of Computers
Deputy for Engineering & Technology

ABSTRACT

The Report embodies the latest results of a continuing research effort directed toward the development of management guidelines, standards, and techniques in the field of computer programming. The work is based upon earlier studies at System Development Corporation that included: the definition of variables affecting computer programming costs; the design of a questionnaire as an aid to collecting data on completed jobs; and the exploratory statistical analysis of 27 completed computer programming jobs to develop preliminary cost-estimation relationships. The present Report is focused upon the statistical analysis of 74 completed computer programming jobs in terms of their resource-costs and related variables, e.g., man months, computer hours. The primary results developed in this analysis are: indices of job difficulty, job type, development environment, and job uniqueness; a "costliness" factor that permits programming tasks to be ranked in this respect; weighted composites of the indices for estimating the cost of particular programming jobs; and scoring and confidence-band techniques for blending intuitive managerial judgments with the formal cost-estimation procedures. Supplementary findings include indications of the relative sensitivity of job cost to changes in the values for the indices, and preliminary comparisons of resource usage between programs produced in machine-oriented or procedure-oriented languages. Also, recommendations are made for future research, including: the collection of more accurate and current data on programming jobs during the production cycle, and the development of a census of computer programming, to enable the design of precise sampling experiments for subsequent analyses.

CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

SECTION I

SUMMARY OF THE REPORT

This Report presents the latest results of a continuing research program directed toward the development of improved methods of estimating, collecting, and controlling costs for managers of computer programming activities. This work has been sponsored since March 1964 by the Electronic Systems Division, Air Force Systems Command, and is part of the Programming Management Project, which was established at System Development Corporation in 1962. The Project seeks to provide management guides, standards, and techniques in the field of computer programming.

The chief results of the work reported here are equations to estimate the resource-costs of computer programming, i.e., man months, elapsed time, computer hours, and number of new instructions. The work to derive these equations follows the pattern of an earlier study and uses statistical analysis of experience data obtained from completed programming projects. Specifically, over the past year, an improved version of the questionnaire that was developed for the previous analysis was used to survey 74 completed programming projects, once again from System Development Corporation. Of the 74, 23 were carried over from the earlier study, after additional information was obtained to insure equivalence of the data. The data base now represents such diverse applications as command and control, utility systems, management accounting, compilers, and research programs, and ranges from efforts of 1 to over 1600 man months.

As before, well-established techniques such as correlation analysis were used to eliminate variables with spurious values and those that showed little predictive potential. Multivariate regression procedures were again applied to obtain estimation equations for the same resource-costs that were considered previously in an earlier analysis, i.e., man months, computer hours, new instructions, and months elapsed. The data upon which the analysis was based reflected the program design, code, and test phases of the production process. System design, program installation, and maintenance were not considered.

To provide techniques that will be useful to managers, the analysis was intended to select predictor variables that were known or could be estimated with reasonable reliability early in the program design phase, e.g., the number of subprograms. Eleven such predictors were defined and grouped into four indices that tend to characterize the programming task: the Uniqueness Index, the Job Difficulty Index, the Development Environment Index, and the Job Type Index. These indices were used as predictors in multivariate regression analysis to obtain estimation equations for various resource-costs: man months, computer hours, new instructions, and months elapsed. In addition, factor analysis procedures were used to define an aggregate of resource-cost variables that can be used to compare and rank programming jobs as to their

"costliness," and an estimation equation using the indices as predictors was obtained.

We expect these indices to remain relatively constant from one analysis to another in a conceptual sense, although their components and weighting may change. Moreover, the use of indices provides a new and intuitively appealing way to compare and rank programming tasks as to their uniqueness, difficulty, development environment, and type.

A novel application of the Stanine scoring technique to the construction of confidence bands was also used to facilitate the blending of managers' intuitive and experiential judgements with the estimates computed from the regression equations.

We regard the development of task indices, their use as predictors in estimating equations, and the Stanine scoring techniques as the major findings of this analysis, and as the elements of a practical management tool for estimating and comparing computer programming resource-costs. At the same time, we consider these results transitional in the sense that they are based on only 74 programming projects from one organization. We are now taking steps to mitigate this constraint by collecting a variety of new data from a number of government and industrial organizations for analysis in late 1965 and early 1966. In addition, plans are being made to design a prototype cost collection system for experimental implementation in 1966 to provide more accurate and reliable data for analysis than is possible with questionnaire methods.

In addition to the development work leading to the major findings, the Report also includes a discussion of two supplementary investigations. In one of these, the task indices were analyzed to determine their relative impact on three resource-costs: man months, computer hours, and months elapsed. This study of cost sensitivity is a forerunner of future work aimed at providing managers with guides to making planning decisions with respect to prospective programming jobs. The other supplementary investigation concerned the effects on program production resource-costs of machine-oriented languages (MOLs) and procedure-oriented languages (POLs). The comparisons were made in terms of three ratios, Production Rate (net instructions/total man months), Computer Usage Rate (total hours/net instructions), and Documentation Rate (total pages/net instructions), and tended to confirm some prevalent opinions that procedure-oriented languages are less expensive in resources. (These results are preliminary, however, since only 14 POL programs were considered.)

In summary, while substantial refinements are anticipated with additional and more representative data, we believe that the second analysis of data from System Development Corporation, as presented in this Report, provides the conceptual foundation for the estimation of resource-costs by computer programming managers. As in our previous work, we recommend that the regression

equations and Stanine plots be used as checks and guides against estimations made by other means, and we solicit constructive feedback as to their effectiveness. We do not, however, consider the estimating procedures described in this Report as sufficiently precise and reliable to be used as the sole basis for management decisions with respect to computer programming.

(Page 4 Blank)

SECTION II

AN INTRODUCTION TO THIS REPORT

This Report is based on research that was performed as a part of the SDC Programming Management Project, under the sponsorship of the Electronic Systems Division, Air Force Systems Command[1], and represents a continuation of the work begun in March 1964, and published in the TM-1447 series of documents.

The Report records the latest results of a continuing effort to apply the rational techniques of management science to one portion of the information processing field--the process whereby a computer program is designed, coded, and tested--and, in so doing, begins to augment recollection with data, and intuition with analysis. The approach used to date involves the collection of detailed information on completed computer programming projects by questionnaire, and the analysis of the results using well-established statistical techniques.

The work is aimed primarily at managers of computer programming activities, who are faced with the problems of predicting, measuring, and controlling economic (and technical) performance and value. While we do not expect that all managers will benefit equally, many will find the relations that are developed between various aspects of computer programming to be useful. Other managers, whose experience and intuition as yet exceed our results, may find quantified support for the relationships they have long felt to exist.

It is clear, however, that the techniques developed herein will be meaningful only to the extent that they are validated in practice. Toward this end, we encourage the use of this Report as a guide to management decision-making and solicit constructive comments from any source. Moreover, we will gladly forward on request the questionnaires and supporting material by which a manager's estimates of a programming task, and detailed information about the actual production process, can be collected and added to our data base, to be reflected in later analysis.

---

[1]Other past and current research by the Programming Management Project has been supported by the Advanced Research Projects Agency (ARPA), the Office of Naval Research (ONR), the SHAPE Technical Centre (STC), and System Development Corporation. To the extent that these parallel efforts in the general area of programming management contributed to the ESD-sponsored work reported in this document, the organizations cited deserve recognition.

To facilitate the user's retrieval of information we have grouped the Sections into the following parts:

(1) A Summary of This Report

(2) An Introduction to This Report

(3) The Purpose of This Research

(4) An Overview of This Analysis

(5) Directions for Future Research

(6) The Technical Procedures for This Analysis

(7) The Major Findings of This Analysis

(8) Application of the Major Findings to Cost Estimation

(9) Supplementary Findings

(10) Appendices

(11) References and Bibliography

The reader is encouraged to turn directly to those portions of the Report that are of immediate interest. While the entire document presents a comprehensive view of the research, each part has been written to be interpretable separately, insofar as possible:

a. Readers who are mainly interested in a broad perspective of the work of the Programming Management Project are referred to parts (2) through (5), above.

b. Readers who are concerned with the analytical techniques used will find these discussed in part (6), above.

c. Readers who are primarily interested in the major findings of this Report, and their application, are referred to parts (7) and (8), above.

SECTION III

THE MANAGEMENT OF COMPUTER PROGRAMMING:

THE STATE-OF-THE-ART SUMMARIZED

While the public image of a manager has undergone a substantial transformation in recent decades, there has never been much doubt that the ultimate standard by which most managers are judged, and by which they judge themselves, is the economic performance of the activities for which they are responsible. It is the ability to motivate, predict, measure, and control economic performance that underlies the functions that traditionally have been given the manager to discharge.

The immediate problem facing managers of information processing activities today is that reasonably precise and generally accepted standards and techniques by which the economic performance of computers and their applications can be measured are nearly nonexistent. Electronic data processing is new and different enough that most traditional management-accounting guidelines and measures are not very meaningful. Decisions are made every day, but they are too often based on an ad hoc blend of experience, intuition, and temerity. While these qualities are certainly essential to some degree, they can be considered adequate only in the absence of well-founded methods.

The information processing manager's dilemma is further compounded by the frenzied pace of computer technical development and application, which shows no signs of slackening. Although their projections vary, experts in the field are in complete agreement that the production and installation of computer hardware and the associated programs have become very big business, indeed, and seem destined to become larger still. Estimates from various industry and government sources indicate that national expenditures for computer programming alone will range from $3 to $7 billion annually by 1970.(1)

It is only natural that the prospect of expending such enormous sums for computer systems has increasingly focused attention on the managerial questions of performance, quality, and effectiveness. This is especially the case with the Federal Government, which is by far the largest single user of electronic data processing equipment and services. Several significant studies along these lines have been issued recently by the Comptroller General, the Bureau of the Budget, and Committees of the House and Senate.(2)

It is generally recognized, however, that these reports represent more of a first step and identification of problem areas for future work than definitive solutions. A great deal needs to be done before the present profusion of information processing terminology and techniques have been analyzed and measures and standards synthesized that are useful to managers in the discharge of their day-to-day responsibilities.

The difference between the state-of-the-art in computer programming management--which is the principal interest of the research presented in this Report--and that which prevails in more mature industries can be perceived most clearly by contrasting the perspective of a potential buyer of a computer program with that of, say, an automobile.

First, an automobile is a tangible piece of equipment whose qualitative attributes can be assessed rather easily. A computer program, on the other hand, is delivered as a set of documents, cards, tapes, and listings, all representing an operational entity that cannot be seen, heard, smelled, or kicked.

Furthermore, the potential buyer of an automobile can make use of the evaluations of popular journals in the field, descriptive literature from the manufacturers, or subjective reactions of friends who "own one." Most of these facts and opinions will be phrased in terms of measures common to automobiles, such as horsepower, turning radius, comfort, style, safety, optional accessories, freedom from repair, and so on.

One who is considering the purchase of a computer program has a much more difficult task in establishing any reasonable criteria on which to make comparisons. Few programs are ready-made; most are tailored to the needs of the user. There are almost no standards for comparing the characteristics of programs with their expected performance. Usually, the design of a computer program is based upon a description of the job to be done, but most often the characteristics are highly qualitative rather than quantitative.

Now, suppose that these contrasts between products are seen through the eyes of a production manager rather than a potential buyer. The manager in the automobile factory benefits immediately from the nomenclature and procedures that are standard throughout the automotive industry. There are techniques available to predict and measure the performance of the product, on both a unit-by-unit and a sampling basis; standards against which the output of both men and machines can be gauged; and an abundance of historical material in terms of which the efficiency of alternatives can be estimated.

Because most computer programming projects are thought of as "one shot" events, the manager of such an activity is faced with an almost total lack of standard measures for the performance or quality of products or tasks, or predictive techniques for estimating the costs and manpower that will be required. As a manager tries to meet estimated budgets or schedules for a programming job, however arrived at, the most common "balancing" factor is quality. Since there is no recognized and reliable yardstick, quality can be compromised in favor of costs or schedules. In addition, he generally has no body of historical data on which to rely in selecting alternative courses of action. Clearly, the computer programming manager would benefit immensely if the kinds of management measures, standards, and techniques that are common to many other

industries, such as the automotive, could also be developed for the field of computer programming.

The evolution of such managerial aids for computer programming has been hindered by several obstacles:

a. The most apparent is simply the youth and technical turbulence of the computer programming field. So much has come about so rapidly that comparatively little time has been spent on synthesis or management research.

b. There is a general lack of agreement on the concepts and terminology in use throughout the computer programming field. While there are many glossaries, such as those prepared by the Association for Computing Machinery (ACM), the Bureau of the Budget, and The American Standards Association, as well as the forthcoming IFIP/ICC Vocabulary of Information Processing,(3) they are not, as yet, generally relied upon, and most organizations evolve their own set of working definitions.

c. Little attention has been given to the definition of attributes that characterize the nature or quality of a computer program as a product. For example, programmers use such terms as "maintainability," "tightness of coding," and "flexibility," but there seem to be no widely accepted criteria by which similar programs could be compared in terms of these factors.

d. Present cost-collection criteria seem to be designed primarily for legal-accounting purposes. For this reason, the historical data that remain after a programming project has been completed are not readily amenable to analysis in terms of the broader needs of managerial planning and control. Generally speaking, except for certain limited types of cost data, historical information is not kept in any organized and cohesive fashion at all. What records remain are not usually comparable from one location to another and, often, not even within different portions of the same organization.

e. Many of the design, schedule, and resource constraints that impinge upon the computer programming process are not well enough understood to be susceptible to meaningful quantification. The interrelationships between the steps in the program production process itself are not well defined, and no general agreement exists on how individual variations should be combined to provide conclusions that are meaningful in terms of the program end-product.

SECTION IV

RESEARCH INTO THE MANAGEMENT OF COMPUTER PROGRAMMING:

MAJOR PROJECT ACHIEVEMENTS AND AIMS

In summary, there is an urgent need for ways of making what in current parlance are called "cost-effectiveness" decisions with regard to computer programming specifically (and information processing generally). But the ability to measure cost effectiveness presupposes reasonably precise and generally accepted measures of cost and measures of effectiveness, which, in turn, presupposes a pool of structured and comparable historical data from which such measures can be derived rationally; this, finally, presupposes the commonly-held notions about concepts, processes, and terminology that would allow such a data base to be collected. As we have observed previously, none of these prerequisites have been more than minimally satisfied.

For more than three years, the Programming Management Project has been working at each of the three links in the research chain leading to measures of cost-effectiveness for computer programming:

a. At the most elementary level, we have modeled the development of computer programs as a process, and dissected it to define a common sequence of tasks. (The most recent result of this effort has been a detailed Planning Guide for Computer Program Development(4).)

b. We have developed and refined a detailed questionnaire, and used it to collect a substantial body of historical data, both quantitative and qualitative, from a wide range of completed programming projects. (Although the data were originally confined to SDC experience, questionnaires are now being completed by a variety of Air Force and industrial organizations that do significant amounts of computer programming.)

c. We have applied well-established statistical techniques to the resulting data base. These have led to the selection of cost factors that seem especially significant, and that form the basis for experimental cost collection systems.

d. We have developed equations for estimating computer programming costs through the design, code, and test phases. These equations, and various other methods of comparing observations with experience are available to managers for use on an experimental basis.

e. We have engaged several of the major accounting/consulting firms to survey the state-of-the-art of computer programming management in government and industry, and to suggest techniques to synthesize the research results with accounting practice.(5)

f. We have done some preliminary work toward the development of management standards in terms of which the performance and quality of programming products can be measured.

We see these efforts as a long-term attempt to evolve products of two kinds: first, a generally accepted and applicable way of collecting resource-costs on computer programming projects, for managerial instead of legal record-keeping purposes; second, standards, guidelines, and techniques that will permit the manager to use the statistical distillation of past experience to predict and control the performance of computer programming activities within his jurisdiction.

By the end of Fiscal Year 1967, we hope to have progressed to the point where results of these two types can be made available on an "operational" basis for the first time. However, it is doubtful that these research products will be definitive or final by the end of Fiscal Year 1967, or even later. There is far too much innovation taking place in the computer programming field, e.g., procedure-oriented languages, time-sharing, etc., to allow much hope along these lines.

On the other hand, the progress that has been made in other fields does suggest that, while managerial problems are not usually susceptible to precise solution within specified periods of time, substantial improvements can be made if a rational approach is combined with practical experience in a step-by-step way.

Our practical object in this work is, quite simply, to help the computer programming manager do a better job--better tomorrow than yesterday, and better in 1966 than in 1965. At the present state-of-the-art, it is only natural that no single technique or result will serve all purposes or solve all problems. A variety of approaches, some more elegant, some less so, will need to be developed along the way. But the primary criterion of our work is usefulness, and of that, managers of computer programming must be the final judges.

The growing awareness of the need for improved management standards and techniques in computer programming (and information processing generally) has also stimulated a variety of work in these areas. In this context, we view our research as one of a community of related projects, all focused upon one or more EDP management problems. Some of the more notable efforts involve:

a. The National Bureau of Standards, which has recently formed a Computer Science and Technology Center. The plans for the Center include work to guide government agencies in the development, measurement, and test of standards for equipment and techniques, including programming languages.

b. The office of the Director of Defense, Research and Engineering (DDR&E), Department of Defense, in which a number of individuals, especially Dr. James Ward, have long been active in EDP management problems.

c. The Electronic Systems Division (ESD), of the Air Force Systems Command (AFSC), which has sponsored a variety of exploratory studies in this area by Auerbach Corporation, Planning Research Corporation, and others.(6)

d. The MITRE Corporation and the System Development Corporation, under sponsorship of Electronic Systems Division, have been engaged in a program to delineate configuration management procedures for computer programming.(7)

e. The Air Force Systems Command has been developing a Cost Information System (CIS), as a part of its Cost Management Improvement Program. The aim is to facilitate the collection of comparable and maximally meaningful cost data for all projects, including computer programming, under AFSC sponsorship. McKinsey and Company provided consulting support to this work.

f. Various other individuals and organizations, such as Brandon and Associates, the Diebold Group, the Standards and Procedures Association, the Data Processing Management Association, and the major auditing/consulting firms, have also been active in this field.

SECTION V

THE OBJECT AND APPROACH OF THIS ANALYSIS

The first exploratory statistical analysis of historical cost data from completed computer programming projects was undertaken in 1964-5(8), and had several aims:

a. To define the major variables that affect the cost of computer programming.

b. Using a questionnaire based on these variables, to collect quantitative and qualitative data for a sample of completed computer programming projects.

c. To demonstrate the usefulness of certain statistical techniques, especially multivariate regression, as a means of exploring the relationships between potential predictor variables that describe the programming job, (e.g., the development environment, the staff, etc.), and project resource-costs (e.g., man months, computer hours, etc.).

A total of 27 completed programming projects, all from the System Development Corporation, were analyzed, based on questionnaire-acquired data. Considering the almost complete lack of comparable historical information on computer programming available when the work began, the research along these lines was quite successful:

a. Over a hundred variables affecting computer programming costs were identified.

b. A comprehensive data-collection questionnaire was developed, and refined by application in the field.

c. A small but diverse and relatively unique body of information on programming efforts was gathered and analyzed.

d. Linear regression equations were derived to estimate some of the major cost parameters, e.g., man months, computer hours, etc.

On the other hand, the derivation of meaningful results from such a small, experimental sample required that some rather severe compromises be made during the analysis:[2]

---

[2] The comments that follow are mainly our self-criticism of the work, but also generally reflect the constructive feedback from our readers. The emphasis is on the analysis of the first SDC data collection.

a. Since effective regression analysis generally requires substantially more data points than variables (the opposite of the situation that initially existed), about two-thirds of the original 100 variables had to be eliminated--some on the basis of statistical relationships, some intuitively--before formal analysis began. However, the winnowing process left much to be desired as to technique.

b. The remaining independent variables were not easy to group as measures of the program production task, e.g., the interrelation between individual predictors such as the size of the data base and the number of document types, on the one hand, and computer hours on the other, as an indication of total program cost, was not apparent.

c. Again, since so few data points were available, no thought could be given to subdividing the population by types of programs. All the data points were thrown into a common pool and analyzed together.

d. The small number of data points also eliminated any possibility of exploring nonlinear regression relationships directly, at least in a statistical way. Effective nonlinear regression analysis generally requires an even higher proportion of data points to variables than does linear analysis.

e. The fact that the few available data points tended to be clustered at the ends of a rather wide range of values, and that exponential relationships between many of the cost and predictor variables appeared to be present, required the elimination of several extreme outliers, and the transformation of several variables into logarithmic form.

f. Finally, the indecisiveness of many of the statistical relationships concerning the predictor variables required the use of "Number of New Instructions," itself an unknown, as a predictor. While the number of instructions can perhaps be estimated more easily in advance of program production than can man months, computer hours, etc., the approach taken was not wholly satisfactory from a practical point of view.

The objectives of the present analysis were, quite simply, to go beyond the first effort in several ways:

a. By using an improved questionnaire to collect data.

b. By gathering a larger base of SDC data, more amenable to statistical evaluation.

c. By developing techniques that would tend to overcome the shortcomings of the first analysis, e.g., meaningful ways of aggregating predictor variables.

These aims were broadened further when the granting of an Air Force Systems Command Report Approval Number provided the authority to collect data from a

variety of military and industrial organizations(9) for analysis in late 1965 and early 1966. As a result, the second analysis of SDC data also became a test vehicle for many of the concepts and techniques that were developed with the military and industrial data bases in mind.

From more than 100 SDC programming projects initially under consideration (including the 27 used in the first analysis), a total of 74 were included in the final data base for this analysis.[3] These represented a variety of programming tasks including management data processing, command and control, research compilers and utility routines, information retrieval, etc.

Although the 74-point data base was nearly three times as large as that available for the preceding analysis, the total number of points was still considered too small for a major effort to be expended on a study of sub-populations. Instead, every effort was made to retain all valid data points in the analysis and thus to develop relationships that seemed to be meaningful for many types of computer programming, and the techniques that would be applicable for the study of later, larger data collections. For example:

a. Only 14 of the 74 data points were coded in a procedure-oriented language (POL), a sample too small and diverse to permit predictive relationships to be inferred with any substantial reliability. Rather than eliminate the POL-coded points, we used the number of machine-language instructions after compilation as a variable that seemed reasonably comparable to the number of machine-language instructions that resulted when the other 60 points were coded in a MOL[4], and analyzed all 74 points on this basis. The price paid for this compromise, of course, was the increase in variation that the POL points contributed to the total data base, which made estimation more difficult.

b. For most of the major variables, e.g., man months, computer hours, words in data base, etc., the few large data points were greatly outnumbered

---

[3] Projects were eliminated from further consideration for a variety of reasons, including dates for availability of the data, low reliability of the responses, etc.

[4] We realize that the number of machine-language instructions after compilation would have been somewhat less if the programs in question had been coded in machine-oriented language directly. Some experts say that, in the case of JOVIAL, this increase in "tightness" varies from 10 to 15 percent. Once again, we chose to accept the increase in variance rather than introduce further uncertainties at this time. Evaluations of the effects of POLs on program production will be considered in future studies.

by the relatively smaller programming projects. To increase the cohesion of the data base, we transformed most of these variables logarithmically rather than eliminate the cutliers.

Within the major objectives of deriving useful relationships from a single, representative data base, the major technical emphasis for this analysis was placed on the grouping of significant variables in ways that seemed intuitively and statistically meaningful. However, the derivation of resource-cost predictors for computer programming is necessarily a long, iterative analytical process, which involves many changes in the variables that are considered and their relative weighting. Sequences of many variables, which change from one analysis to another, may not be intuitively attractive to the user.

On the other hand, although information on many variables is collected, most reflect, perhaps, a half dozen major aspects of the programming job: the difficulty of the design and coding, the development environment, the staff, the equipment, and so on. It is much more meaningful to think in terms of indices of these various aspects of the programming that will tend to remain reasonably constant from one analysis to another, although their composition and weighting may change. Such indices will also allow ranking or scoring of programming tasks as to difficulty, type, etc., based on demonstrably meaningful composites of significant variables.

We have developed four such task indices in this analysis, and have used them to estimate computer programming resource-costs. Another grouping of cost variables was also obtained to allow the comparison of programming efforts as to their "costliness." We believe that these aggregates of variables, which are discussed in detail in Sections XV and XVI, are a more practical tool for management use than was available previously, and consider the estimating equations presented in Section XVII as the major finding of this Report.

SECTION VI

THE TECHNICAL PLAN FOR THIS ANALYSIS

1. Background. It is a fact of statistical analysis that an abundance of quality data allows the researcher comparative freedom as to techniques. On the other hand, the most refined methods can hardly compensate for a disproportion of variables to data points, as was the case in the preceding study of this series. Although a total of 74 data points is not sufficient for all analytical purposes, it does cross the threshold into the realm of meaningful inferences about the computer programming process. The technical plan that was developed for this analysis reflects both the larger, more representative base of data and the broader, more ambitious objectives outlined previously. To clarify the presentation, the research results are discussed separately, in Sections IX through XX.

It should be noted that the Technical Plan is simply a guide to insure that the analyses of different batches of data are comparable, and that no fruitful steps are neglected. The analytical process may dictate the skipping of one step or another, or the iteration through a certain procedure several times. In this sense, the Technical Plan is simply an analytical framework, a checklist, in terms of which the evaluation of the data can be conducted.

2. Outline of the Technical Plan. In general terms, the development of a cost prediction model for a single population of computer programming projects--the main object of this analysis--was carried out in accordance with the following sequence:[5]

a. Data Quality Control, Correction and Formatting:

(1) Validation, error correction, etc.

(2) Analysis of outliers, modification of data.

b. Initial Definition of Dependent and Independent Variables

c. Formal Analysis:

---

[5]The technical plans for the other, smaller studies that were conducted in parallel with the main analysis--e.g., development of resource-cost ratios, and MOL versus POL comparisons--are presented together with the particular results, in Sections XIX and XX. The technical plan for the study of subpopulations will be developed in subsequent reports dealing with these matters.

(1) Scatterplot analysis of independent versus dependent variables.

(2) Correlation analysis of all variables.

(3) Factor analysis of independent and dependent variables.

(4) Regression analysis of dependent factor measures.

(5) Intuitive modification of factor measures.

(6) Development of independent variable aggregates (indices).

(7) Regression analysis of independent variable aggregates (indices) against dependent variables.

(8) Regression analysis of dependent variables and factors against independent variables, factors, and indices.

d. Analysis of Regression Model-Fitting Errors:

(1) Exclusion of undesirable data points.

(2) Application of curve-fitting techniques other than the least-squares method.

(3) Search for overlooked variables, factors, or indices.

3. Discussion of the Technical Plan. Although a detailed review of the theory underlying each step of the Technical Plan is beyond the scope of this Report, and can be found in the Bibliography, a brief exposition follows:

a. Data Quality Control, Correction, and Formatting:

(1) Validation, error correction, etc: The particular statistical techniques used--factor and multivariate regression analysis--are especially sensitive to spurious data, e.g., from misinterpretation of the questionnaire, transcription errors, etc.

Prior to the beginning of formal analysis, all data are checked for accuracy and quality.

(2) Analysis of outliers, modification of data: Ideally, the data for each variable should be continuously and normally distributed, and the data points should be independent of one another. Since these conditions are seldom met, a certain departure from the ideal can be tolerated. The closer to the ideal, however, the more trustworthy will be the statistical results. (Although variables with discontinuities and very skewed distributions are

avoided whenever possible, they may be included, on occasion, to permit early state-of-the-art results.) The biasing effects of pronounced skewness can usually be mitigated by transformation, e.g., to logarithmic values.

Apart from poor distributions, some variables may include data that are somewhat outlandish with respect to the median. Since such data have a disproportionately degrading effect on factor and multivariate regression analyses, which deal with weighted values, it is common practice to apply various statistical measures to determine whether or not the outliers are part of the same population. A variety of rules are available to facilitate this determination, e.g., those by Dixon, Grubbs, Murphy, Thompson, and Tukey(10). Particular extreme data are usually replaced by a value closer to the median of the variable, i.e., Winsorized, to avoid an undue prejudice of the subsequent analysis. In some cases, if many of the values for the variables that describe a single programming effort are extreme, the data point may be deleted.

b. Initial Definition of Dependent and Independent Variables. The variables are separated into two logically distinct classes: those to be predicted, and those which are to serve as predictors. When historical data are being analyzed, care must be exercised to insure that predictor variables will be available in measurable form or can be estimated reliably, when the prediction equations are to be used.

4. Formal Analysis:

a. Scatterplot Analysis of Independent and Proposed Dependent Variables. Visual analysis of the relationship between pairs of independent and proposed dependent variables often reveals special relationships, such as nonlinearities or potentially spurious outliers, that are difficult to determine otherwise.(11)

b. Correlation Analysis of All Variables. Correlation analysis involves the study of the intercorrelations between all the variables, i.e., the correlation matrix. Although factor and/or multivariate regression techniques are more powerful and are relied upon for the main analysis, the correlation values themselves provide insights that could be overlooked otherwise.(11)

Two types of relationships are especially important: first, that the intercorrelations between the independent variables are as small as possible, i.e., that the predictors are maximally independent; second, that the intercorrelations between each independent and the dependent variables are as large as possible, i.e., that the so-called "validity" of each independent variable be maximal with respect to the dependent variables.(11)

c. Factor Analysis of Independent and Dependent Variables. Technically, factor analysis is a method for studying the intercorrelations among a group of variables to define the common influences that are present. The specific

aim is to reduce the dimensionality of the intercorrelation matrix from one composed of variable-versus-variable correlations to a smaller, more easily interpretable one composed of factor-versus-variable correlations. The principal components model first derived by Hotelling(12) and the varimax factor rotation procedure developed by Kaiser(13), are well-established techniques for dealing with a problem of this type.

d. Regression Analysis of Independent Factor Measures. To obtain factor measures, the multivariate regression process is applied to secure the best combination of variables for defining each factor. In this process, the factors become weighted composites of the component variables.(14) (Since the factors have no known mean and standard deviation, arbitrary values are assigned, e.g., in this Report, a mean of 100 and a standard deviation of 20 have been found to be convenient.)

e. Intuitive Modification of Factor Measures. In using factor analysis, there is no guarantee that the combinations of variables derived will be intuitively meaningful or interpretable. It is always possible and sometimes desirable to modify factor analysis results in the light of practical considerations. The value of engaging in this sort of adjustment depends, of course, on the ingenuity of the analyst. When there are strong practical considerations for making a modification, however, the change should be made.

f. Development of Independent Variable Aggregates (Indices). It is possible to group independent variables solely on the basis of intuition, and to weed out those whose intercorrelations with other predictors in the group are higher than their correlation with cost. The result is the opposite of a factor: a combination of highly predictive variables that express intuitively similar but statistically different parts of the variation, rather than reinforcing the same underlying contribution.

g. Regression Analysis of Independent Variable Aggregates (Indices) Against the Dependent Variables. As in step "4d," regression analysis is applied to obtain the weighted composites of each index that are most effective as predictors.

h. Regression Analysis of Dependent Variables and Factors Against Independent Variables, Factors, and Indices. Once the dependent and independent variables have been defined, the primary aim of multivariate regression analysis is to provide a rational basis for the selection and weighting of the predictors.(14)

A nonlinear regression approach may be applied, depending on the nature and amount of data and the relative advantage of this technique.

i. Analysis of Regression Model-Fitting Errors

(1) Exclusion of undesirable points: A desirable follow-up to the definition of a prediction equation is to evaluate the nature of the variance that is not accounted for by the model. Outlier points can often be identified using bivariate plots of actual versus predicted values. The elimination of such points, when justified on the grounds that the outliers are truly part of another population, will permit an improvement in the confidence of the predictions, based on a reanalysis of the data base.

(2) Application of curve-fitting techniques other than the least-squares method: Although the least-squares method of fitting the prediction line is the most common, other approaches, e.g., equalizing the variance of the observed and predicted values around the estimation line, may lead to a distribution of variations that is more consistent over the values of interest. Often, the loss in predictive power and confidence is small enough to permit an adjustment of this type to be made.

(3) Search for overlooked variables, factors, or indices: Occasionally, a cluster of data points may lie significantly away from the prediction line. This usually suggests that an entire factor, index, or variable, that represents a characteristic common to all the points, has been overlooked in the definition of the model.

5. Application of the Technical Plan. In general, the analysis was carried out according to the plan. Certain steps, however, were left for subsequent analyses. These are:

a. A more intensive application of factor analysis methodology to define additional aggregates was not carried out. This will probably be done when the data from government and industry have augmented the 74-point base.

b. Nonlinear regression analysis was not applied, primarily because the number of data points was insufficient. Also, the transformation of many variables to logarithmic form produced relationships that were essentially linear. Nonlinear regression analysis may be used with larger data bases, if it seems advantageous.

c. The analysis of regression model-fitting errors, step "4i," was not done, primarily due to a lack of sufficient time. While the intent of this analysis was to consider all 74 data points together, the deletion of one or two programs with extreme values could improve estimation confidence for the remainder. (This is not the same as step "2a," which deals with outliers for individual variables.) We intend to repeat this analysis after step "4i" has been done, in a subsequent study.

SECTION VII

AN EVALUATION OF THE PRESENT RESEARCH

As indicated previously, the intent of this analysis was to progress beyond the earlier work and, hopefully, to achieve more meaningful and reliable results. As will be substantiated in later sections dealing with the findings, these objectives have been realized to a considerable degree.

At the same time, there is no denying that, against the whole spectrum of computer programming activities with which the manager must deal, much remains to be done.

For instance, the research to date has not encompassed the so-called system design, system test, and maintenance phases, which fall at either end of the program design, code, and assembly test cycle. Neither has the cost of managerial and support overhead beyond the first level of programmer supervision been included. While these omissions can be attributed to the difficulty of defining generally applicable and comparable ways of measuring these resource-costs, there is no question that a wholly realistic picture of the computer programming process cannot be portrayed in their absence. (Some managers are correct in observing that, in certain cases, e.g., very large, complex program systems with many changes, the phases we have omitted may be more expensive than those that have been included.)

A related problem is that of estimating costs before the program design phase, e.g., before the start of system design. While reliable cost estimation is difficult at any step in the programming process, the earlier the predictions are made, the more hazardous the task.

Another deficiency is the lack of an effective way of measuring the relative capability of the programming staff. While various combinations of average experience and positional rating have been tried in the last analysis and this one, an effective yardstick in this critical area remains to be developed.(15) The organization of the programming staff, e.g., project or "assembly-line" approach, has also not been considered.

Also, there is the matter of machine and language comparability. No meaningful and reliable way has been found to gauge quantitatively the effects that different computers and/or compilers have on equivalent programming jobs. These considerations are being complicated further by the proliferation of new equipment and new techniques, such as language, time-sharing, etc. Although their adequacy and compatibility in terms of our research needs is not clear at the present time, some experimental ranking systems, e.g., Auerbach's vector approach(16), may be useful in this respect.

For these particular shortcomings, we can express at least the hope of alleviation through additional research; but there is the far more fundamental and disturbing criticism that the present data base, for all its diversity and uniqueness, still represents a collection of information that is almost exclusively after-the-fact. Questionnaires were filled out only after a programming project was completed and, what is more, included many queries on which data were not normally collected by the respondents at all, and which could therefore only be answered in a "best-recollection" fashion.

While these after-the-fact data have been quite useful in allowing us to get a handhold on the relationships that affect computer programming activities, the data are clearly not adequate for the development of the reasonably precise techniques that are required for management control. The accuracy of responses in several cases such as Number of Instructions Discarded Due to Errors, is difficult to determine. Moreover, the data do not provide much help in analyzing the variation of resource-costs through the program production process, i.e., the flow of expenses from one phase to another.

To go the next step, to define reliable relationships between the various cost parameters and translate them into techniques that will be useful to managers of computer programming projects in the performance of their day-to-day responsibilities, we know that a substantially more reliable and continuous data base is required. Such a data base can only be gathered within existing cost collection systems, on a regular basis, during the life of a programming project. We call this "on-line cost collection."

The accumulation of costs during program development rather than after-the-fact, however desirable, cannot become a reality until such a procedure is not only meaningful technically but feasible economically. The collection of information itself has a cost; and the notion of increasing the frequency and comprehensiveness of computer program cost data, whatever its analytical potential, is simply not practical until certain prerequisites have been satisfied:

a. The number of variables on which data are to be collected must have been reduced to a minimum;[6] and evidence as to the effectiveness of these variables in predicting costs must have been demonstrated from both the scientific and managerial points of view.

---

[6] Of course, this "minimum" will change as the programming field changes, and new relationships are developed and demonstrated.

b. The variables themselves, and the methods of recording and collecting them, must have been adequately defined and made compatible with the accounting systems and techniques that are generally prevalent, or their logical extensions.

c. Since sampling of computer programming projects is an economic necessity (it is not practical to collect extensive data from all programming efforts), there must be reasonable evidence, from a statistical viewpoint, that the analysis of the resulting data will yield inferences that are meaningful in the broader population of programming tasks.

The impact that these requirements have on the prospects for future research in this area is the focus of the following Section.

SECTION VIII

SOME AREAS FOR FUTURE STUDY

The approach taken in this work is based on the proposition that the analysis of comparable data on the resource-costs of computer programming projects--so long as it is done on a reasonably economical basis--is central to the long-term development of meaningful management standards and techniques in this field.

In the previous Section, we have commented on the shortcomings of the ways in which data were collected for this analysis, e.g., after-the-fact information leaves something to be desired as to reliability and validity. The "on-line" approach promises substantially to alleviate this deficiency, and to provide the method whereby a more comprehensive assortment of information can be collected in parallel with the program production process.

In addition, our present plans will carry us beyond the work in this Report in several ways:

a. The data that are being collected from government and industry will result in a substantially larger and more representative data base, and will likely permit the analysis of separate subpopulations, i.e., types of programming jobs.

b. The results of the analyses of questionnaire-acquired data will be blended with accounting practice and terminology into a prototype cost model for computer programming, which will be experimentally applied to the on-line data collection.

A more fundamental problem remains, however. The mechanics of data collection and analysis notwithstanding, what types of computer programming tasks, in what proportion, should be considered for the on-line data phase? By what criteria is the composition of the analytical data base to be determined?

Practically speaking, no standards were applied in the accumulation of the present data base, apart from questionnaire accuracy and completeness. Managers in various areas of the System Development Corporation were asked to supply representative samples of the programming done under their jurisdiction, but no data points were rejected on the grounds that they were non-representative.

As a result, the data base tends to resemble a cross-section of programming practice, e.g., there are many smaller programs and few larger ones. Since this research is in its early stages, with an object of defining relationships common to programming generally, a data base of this type is considered

acceptable. As the techniques employed herein become more familiar and tested, however, and as the emphasis moves more toward results that are useful to computer programming managers in the performance of their day-to-day responsibilities, some criteria must clearly be enforced as to the composition of the data base from which the end-product relationships are to be derived.

One approach is to collect increasing amounts of data from organizations of all types and sizes that do computer programming. As the data base becomes larger and the sources of information more diverse, the resulting collection should approximate the programming that is being done in the field. In the absence of any criteria other than size, rather massive amounts of data would have to be collected to assure that every type of task was adequately represented--e.g., machine-oriented languages, procedure-oriented languages, on-line usage, time-sharing, jobs of different sizes, complexities, applications, and so on.

Realistically, the time when a representative data base on computer programming could effectively be gathered by the process of "exhausting" the population no longer seems economically feasible, and too much innovation is taking place in the computer field to suggest that data collection problems are becoming simpler in this respect.

The obvious alternative is to rely on some sort of sampling process whereby the whole population of computer programs is adequately represented with the collection of a relatively small amount of data.

Of course, this is not as easy as it sounds. Technically, a random sample (which is what we are referring to) is one in which every item in the population has an equal chance of being selected, and in which the selection of one item or another is independent, i.e., there is no element of collusion in the selection process.

For our purposes, the first restriction is by far the most serious. In the previous analysis and in this one, all the data were gathered from different parts of one organization, the System Development Corporation. We are now collecting questionnaire data from about fifteen government and industry sources. While this is a step in the right direction, we have no reason to assume, for instance, that all computer programming projects in the population have an equal chance of selection. Quite the contrary; we know they do not.

The essential problem is the absence of a standard by which the representativeness of the sample can be judged and, therefore, the extent to which statistical relationships derived from the sample can be relied upon. The way in which this problem has been surmounted in the face of similar difficulties has been to rely on a census. In political forecasting or marketing, for example, the "population" is as complex and dynamic as that of computer programming. Very precise samples are drawn, and inferences made, by defining the

proportion of different classes of data that should be included using a census of the population, and then selecting at random within each category until the correct ratio is achieved.

Fortunately, in political forecasting or marketing, the rudimentary categories are relatively clear and generally acknowledged: there are, for instance, men, women, Caucasians, Negroes, Orientals, employed, unemployed, Republicans, Democrats, city-dwellers, farmers, and so on; and the assumption that these differences are related to voting, or purchasing, seems reasonable. But there are no comparable and generally applicable or accepted categorizations for computer programming. (It is not clear that many of the prevalent distinctions, e.g., real time, on-line, command and control, etc., are sufficiently precise or meaningful for census purposes.)

It is our hope that, out of the research in this Project and other parallel efforts in the field, meaningful and reasonably precise classes of computer programming will be defined and validated over a period of time. Such developments, in turn, will permit the essentials of a census of computer programming tasks to begin to be gathered. While the prospect for quick or easy progress in this area is minimal, we suggest that the direction proposed is essential to the future of effective management research in the field.

For the more immediate future, several other areas, beyond those that are a part of our present plans, seem to merit additional study:(17)

a. A significant and largely unexplained aspect of programming cost estimation and control concerns the management of program changes and reprogramming. This type of work accounts for a large percentage of total programming costs today, and will increase as the number of installations grow. Much of the motivation for modifications or additions to existing programs stems from the frequent introduction of new computers and configurations (e.g., time-sharing), languages, applications, etc. Since both the extent to which changes will be made, and nonessential, but useful, improvements in the program made along with the changes is difficult to forecast, the impact on total costs of the maintainability of the program and the stability of the design have not been given the attention they deserve. Since the programming activities are essentially the same, one could hypothesize that the factors influencing the cost of reprogramming would be the same as those in the present analysis. But additional factors that reflect changes to the existing products, e.g., the data base, documentation, program design, etc., would have to be considered.

b. The lack of quality or performance measures for computer programming products was also cited in the previous study. Present criteria in this area tend to rely on such relatively tangible matters as whether or not the functions noted in the specifications are performed by the program, whether the program operates within the existing time and storage limitations, and so on. For the most part, the question of comparing, let alone predicting, the quality or maintainability of a program product is beyond the state-of-the-art.

c. The effect of the development computer and its configuration on the cost and quality of program production should be thoroughly investigated. At present, most computers are used for both ADP operations and program development. The number of users and applications at a given facility may be quite varied, however; e.g., a time-shared installation, or restricted, e.g., a single-user, single-application military complex. There are several rather obvious areas in which the characteristics of the development computer could lead to trade-offs that affect programming costs. For example:

(1) Procedure-oriented versus machine-oriented languages.

(2) Time-shared versus "closed shop" operations.

(3) Semiautomated support versus additional computer/utility systems.

(4) Centralization versus decentralization of computing facilities.

d. The amount and nature of the documentation that is required to augment the program product has a major impact on total cost. Considerable controversy prevails as to the quantities and types of documents that are preferable. Although some work has been done to systematize programming documentation, less effort has been directed toward determining the cost-versus-value trade-offs of various alternatives, e.g., decision tables, flow charts, key words, prose, etc.

e. Testing of computer programs is often the most expensive, time-consuming, and costly part of the production process. Little has been done to systematize the planning and execution of program tests, and, most important, to provide the manager with guidelines to measure both cost and quality of the result. A related question is the degree to which quality control of design, programming, documentation, etc., can reduce the effort that need be devoted to testing.

f. The initial analysis and design of the information processing system, often called "system design," is a costly part of development that merits additional study. Since the products of this phase are the source documents for the remainder of the programming process, they exert a considerable influence on cost and quality. Although much theory on system design exists, practice appears to differ significantly. There is a need to unify theory with practice and provide comparative measures of quality, value, and cost.

SECTION IX

PROCEDURES FOR THE COLLECTION AND QUALITY CONTROL OF THE DATA

1. The Data Collection Questionnaire. For this analysis, the questionnaire used in the first study (18) was simplified and clarified.[7] The seven divisions were retained:

(1) Summary of Costs
(2) Operational Requirements and Design
(3) Program Design and Production
(4) Data Processing Equipment
(5) Programming Personnel
(6) Management Procedures
(7) Development Environment

However, a number of significant changes were made as a result of the earlier study. For instance:

a. The responses to several questions were felt to be consistently unreliable. This led to deletion in some cases. The "Data Accuracy Index," for example, required the respondent to estimate the accuracy of his information. The results indicated that, although the concept was appropriate, the answers were not statistically useful, and the question was deleted.

b. Certain questions were amplified to provide more meaningful data. The Total Number of Instructions Discarded, for example, was separated according to the reasons for the rejection, e.g., programming errors or operational changes.

c. Questions that seemed intuitively meaningful but subject to misinterpretation were supplemented with a brief definition of the terms causing difficulty. In the case of the system complexity ranking, for example, each of the five levels among which a respondent could choose was briefly described.

d. Questions that revealed little or no statistical significance, or proved to be redundant to other, superior questions, were considered for elimination, e.g., the security classification of the resultant documentation.

e. The format of the questionnaire was improved, e.g., photographic reduction was used for definitions, consistency in appearance was emphasized, etc.

---

[7] This was done by L. Farr and H. J. Zagorski.

The definition of a "data point," and restriction of the questions to the program design, code, and assembly test portions of the production process were retained. The rationale in these areas is given in the report of the previous analysis.(19)

One important obstacle to a more sweeping revision of the questionnaire was the desire to retain, for the second analysis of System Development Corporation data, the 27 points that had been used previously. A supplement to the earlier questionnaire was designed to make the first data base comparable, and each of the original respondents was asked to provide additional information. In 3 of the 27 cases, sufficiently reliable records were not available and the data points were dropped from further consideration.

The questionnaire was used in the form shown in Appendix I. (The questions that were included in the Supplement to the first questionnaire are also noted.)

2. Methods For Data Collection. As in the earlier data collection, no sample design, in the classical sense, was used. Instead, managers throughout the System Development Corporation were encouraged to complete questionnaires for a representative assortment of the programming activities under their jurisdiction. The diversity of activities within the Corporation, and the widespread response, greatly aided our efforts to obtain a well-balanced data base.

Over 100 programming jobs were considered for the sample at one time or another. Various considerations, e.g., tightness of schedules, unavailability of personnel to complete the questionnaire, extension of the test phase beyond the sample cut-off date, etc., gradually eliminated potential respondents until a total of 78 completed questionnaires--including 23 from the first study--were available for formal analysis.

3. Quality Control of the Data Base. Over a period of weeks, each response for each variable was evaluated by the Project staff. Where there was some doubt as to the appropriateness of the answer, e.g., the respondent misunderstood the question, wrote in the wrong numbers by mistake, neglected to provide an answer, etc., the question was flagged for further study. In each such case, the respondent was contacted, if possible, and the particular problems were discussed in detail. In most cases, suspicious answers were resolved by this procedure. In some instances, seemingly spurious data were demonstrated to be valid and were retained.

Despite all efforts along these lines, certain responses could not be justified, either because no adequate records existed, or because the only person who could have illuminated the situation was no longer available. Four questionnaires were dropped from further consideration because the majority of their responses fell into the suspicious category. Certain variables, e.g., frequency of program operation, were misinterpreted often enough that they were eliminated from the analysis for all data points.

Finally, for 81 of about 7700, or about one percent of the items in the data base, when most of the responses for a questionnaire were considered valid and all attempts at clarification of the suspicious or nonexistent information for specific answers failed, values were estimated by the Project staff. In each case, these were intended to be nonbiasing, i.e., average for the type of program involved. The values that were estimated are noted in Appendix III.

The 74 data points that remained for the formal analysis represented a variety of programming tasks, e.g., command and control, compilers, information retrieval, operational utility systems, research programs, management information systems, and others.

SECTION X

THE CHARACTERISTICS OF THE DATA BASE

1. The Age of the Data Base. An important question regarding the data that have been collected is whether they fairly represent the computer programming state-of-the-art, or whether innovations in techniques and the equipment environment have rendered the data obsolete before the analytical results can be applied. While no complete answer can be given, we have made some comparisons on the very conservative assumption that the date on which program design began is an indication of the currency of the data. (In fact, of course, for many programming jobs, one or two years elapse between the start of program design and the completion of the test phase.) On this basis the second sample of data from the System Development Corporation is now significantly more up-to-date than the first:

	First SDC Data Base	Second SDC Data Base
Average Age at Present	3.2 years	2.0 years

By comparison, the average age of the data base that is now being collected from government and industry is only 0.8 years(9).

It is important to note that the average age of the first data base from the System Development Corporation at the time it was analyzed was about two years, the same as the second data base. After the analysis of the government and industry data is completed, the average age of the data will have increased to about 1.5 years, which seems about the minimum lag between the average program design start date and the completion of analyses for the present questionnaire methods of data collection. The on-line data collection techniques described in Section IV are expected to shorten significantly this lag between the state-of-the-programming-art and the availability of the research results.

2. Distribution of the Data. A careful study of the data distribution for each variable revealed that, except for those coded in single digits (e.g., 1, 2, or 3), or percentiles, most were exponentially distributed, i.e., a great many small values relatively close together at the lower end of the scale, and few, rather large and sparsely distributed points at the high end. The fact that effective multivariate regression analysis presupposes a reasonably continuous and normal distribution of data made the mitigation of the exponentially scattered points a major concern.

One approach would have been deliberately to collect enough data on large programming tasks to balance the data base. This would have been difficult, due to the relative scarcity of the more complex and sizeable programs, and the unstructured-sample approach was selected.

Another alternative, which was actually used in the first analysis, was to delete several of the more extreme outliers. However, as indicated in Section V, we did not consider the elimination of data on such grounds to be in consonance with the main objectives of this analysis, i.e., to demonstrate relationships and analytical techniques that were common and applicable to all types of programming tasks. We realized that the development of the most effective management devices required that we reduce the total variation by studying various subpopulations of the data base separately. Since a much larger and broader collection of information was expected with the addition of data from government and industry, and since the 74-point base was considered to be at the threshold of statistical reliability, we chose to postpone the consideration of subpopulations for the time being and use this analysis as a test vehicle for demonstrating the techniques that were to be applied in subsequent efforts. On this basis, no individual data points were eliminated from consideration due to their extreme values.

We would also like to emphasize, however, that the viewpoint that is characteristic of this analysis, i.e., combining all the data into a common base, does not preclude refinement of the results with extreme values removed. We intend to develop these relationships in subsequent reports of this series.

SECTION XI

THE FORMATION AND TRANSFORMATION OF VARIABLES

1. The Formation of Additional Variables. One of the directions taken with this analysis was the development of ratios to indicate relative, rather than absolute, differences, e.g., the variation in computer hours expended for comparably sized programs. Although there was not sufficient time in this analysis to explore these ratios in detail, we expect them to be useful in later studies, especially those of subpopulations. (The ratios that were constructed as a part of this analysis are shown in Appendix II, as variables 109 through 148.)

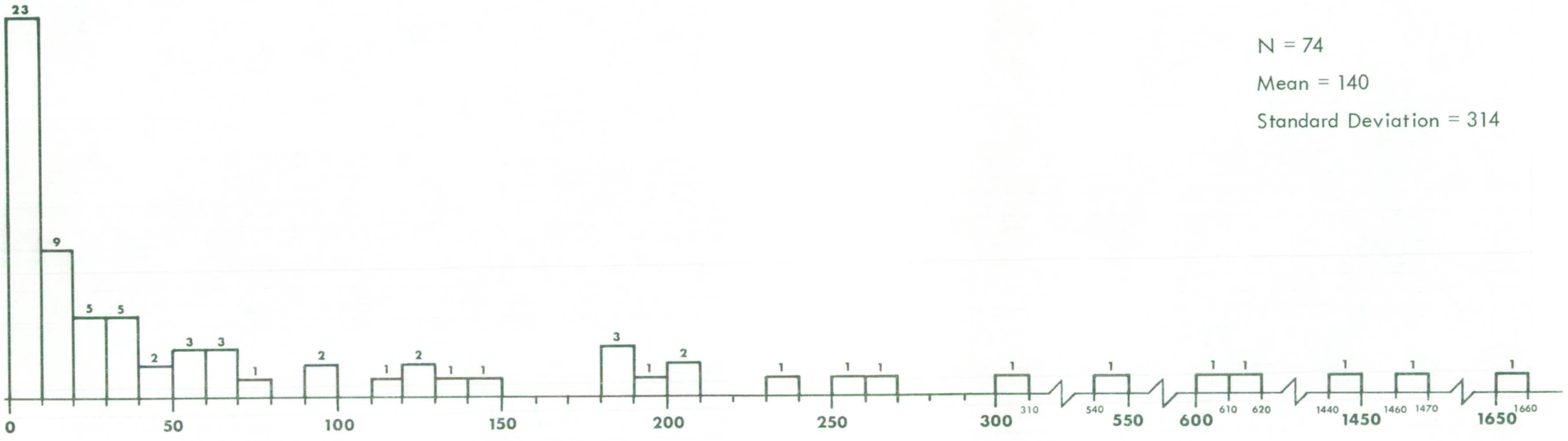
Three of the ratios, Production Rate, Computer Usage Rate, and Documentation Rate were applied to the comparison of the effects on program production of machine-oriented and procedure-oriented languages, and are discussed in detail in Section XX.

2. Transformation of Variables. To compensate for the exponential distribution of many variables, a logarithmic transformation was applied. The effect can be illustrated by Figures 1 and 2, showing the frequency distribution of man-months before and after the transformation.

The logarithmic transformation clearly pulls the extreme values toward the median enough to provide a reasonably symmetric unimodal distribution approaching normality.

The effect of the logarithmic transformation is illustrated further with the following frequency distributions for the variables Computer Hours, Months Elapsed, and Number of New Instructions. (Once again, for comparative purposes, the distributions are shown with a mean of 100 and a standard deviation of 20.)

3. Coding of the Variables. The title, identity number, and coding of each variable used in this analysis is provided in Appendix II.



N = 74

Mean = 140

Standard Deviation = 314

FIGURE 1

FREQUENCY DISTRIBUTION OF UNTRANSFORMED TOTAL MAN MONTHS

(Page 38 Blank)

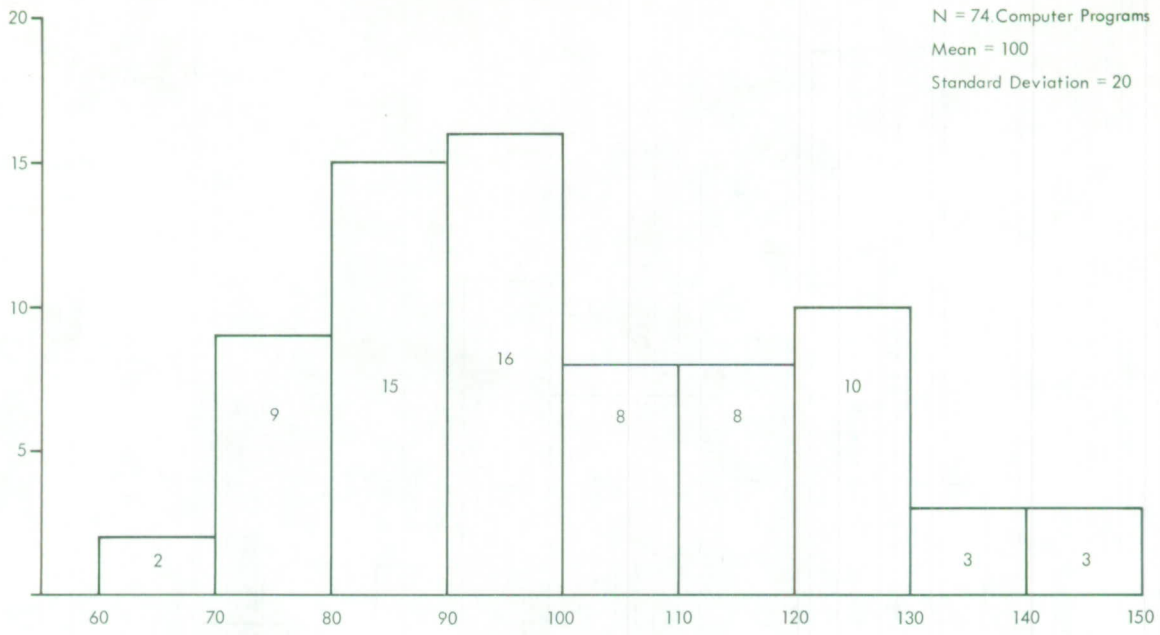


FIGURE 2

FREQUENCY DISTRIBUTION OF LOG_{10} TOTAL MAN MONTHS

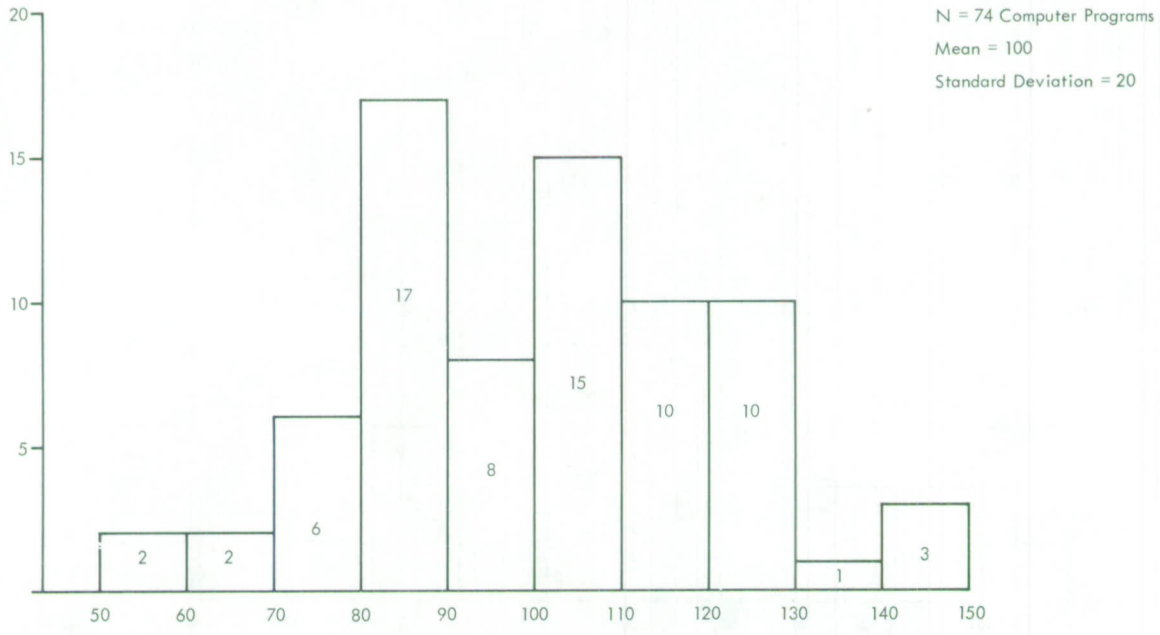


FIGURE 3

FREQUENCY DISTRIBUTION OF LOG_{10} COMPUTER HOURS

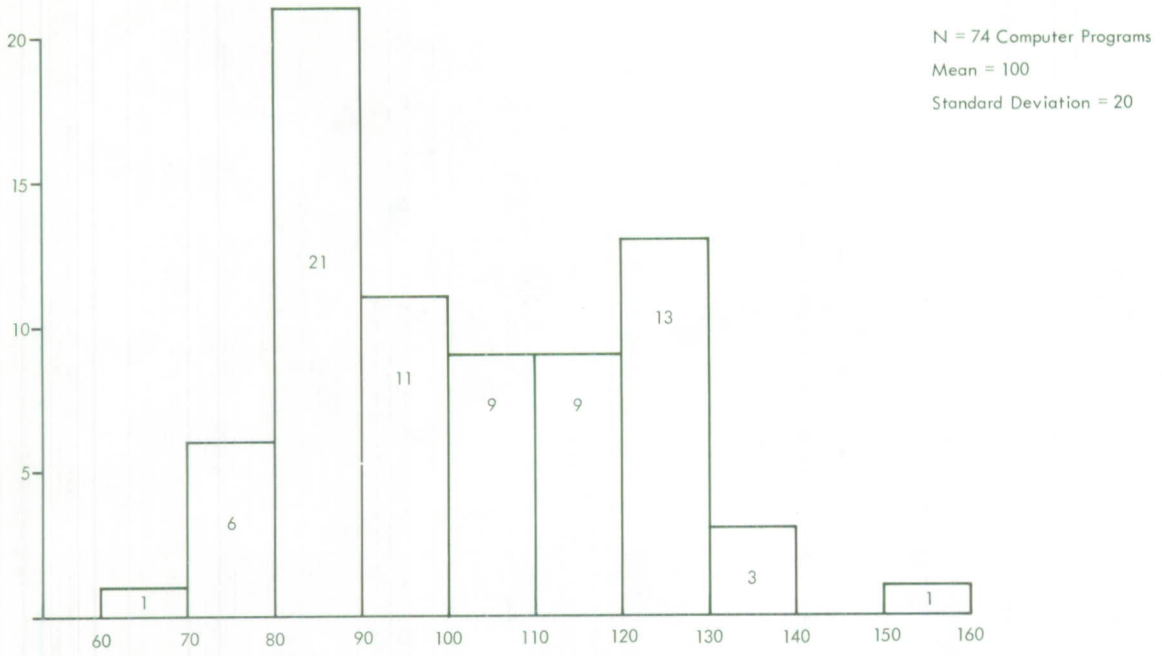


FIGURE 4

FREQUENCY DISTRIBUTION OF LOG_{10} MONTHS ELAPSED

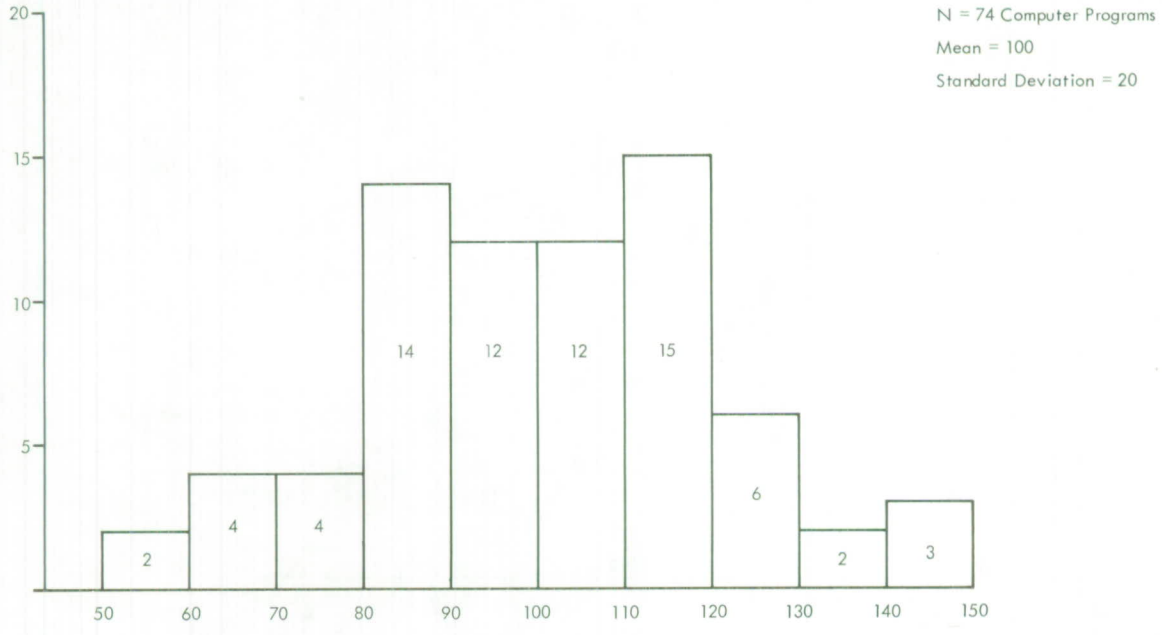


FIGURE 5

FREQUENCY DISTRIBUTION OF LOG_{10} NEW INSTRUCTIONS (MACHINE)

SECTION XII

SCATTERPLOT AND CORRELATION ANALYSIS: IDENTIFICATION OF DEPENDENT VARIABLES

1. Scatterplot Analysis. After logarithmic transformations had been applied to the data where necessary, over 400 computer-prepared scatterplots of bivariate relationships were prepared and evaluated. The primary purpose was to provide visual assurance that the transformed variables were well distributed, e.g., that extreme outliers that could bias subsequent analyses were not present.

Several of the more interesting plots are shown in Figures 6, 7, and 8. To facilitate interpretation, the 67-percent confidence bands, in which that proportion of values would fall for normally distributed data, are shown. Note the obvious linearity of the relationship for the transformed variables (i.e., for raw values plotted on log-log paper). Of course, these plots are for illustrative purposes only; the confidence bands are too wide to be useful for meaningful estimation.

2. Correlation Analysis. The first correlation matrices were composed of 104 rows and columns, which essentially represented the variables taken directly from the questionnaires, after transformations had been accomplished as necessary.

The early eliminations were based on correlations that were either relatively high or low. Since the Standard Error of a null correlation coefficient for 74 data points is $1/\sqrt{74}$ or about 0.12, the search for prediction variables focused on those that exhibited validities of at least 0.20 with likely dependent variables, e.g., man months. Potential predictors that showed intercorrelations above 0.20 were also evaluated for potential redundancy.

In all these cases, correlation analysis was used as a guide to highlight relationships that could bias later analysis. Since regression analysis computer programs with a subsetting feature were available, these were also applied to compare the sequence in which the variables were selected for prediction. The final decisions as to whether a variable should be eliminated from further consideration were made by the Project staff, based on the statistical results and the projected utility of the particular variable.

The original 104 variables were gradually winnowed, using the procedures outlined, until 69 variables remained. These are listed in Appendix V, and the deletions can be determined by a comparison with Appendix II.

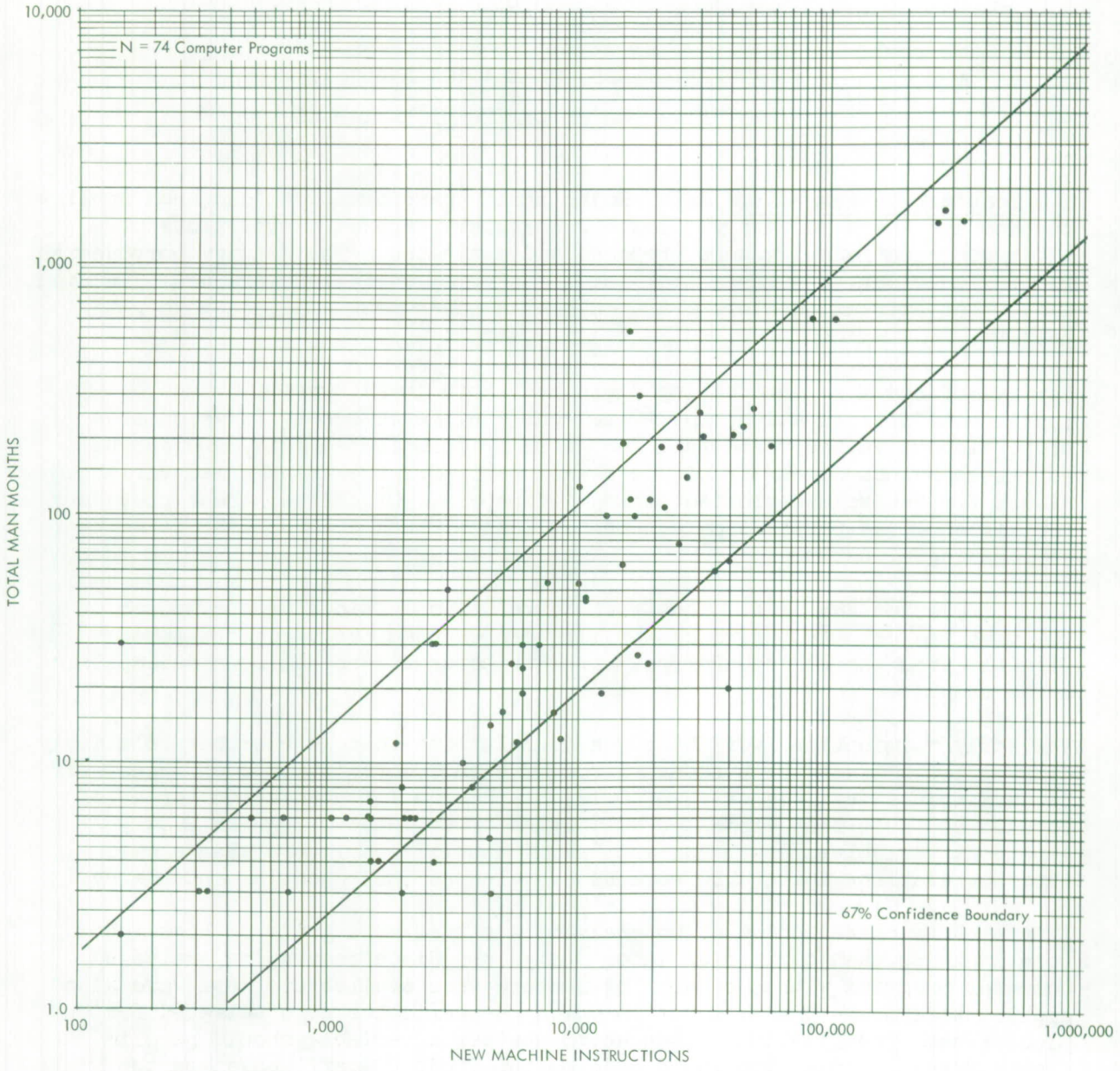


FIGURE 6

RELATIONSHIP BETWEEN NEW INSTRUCTIONS AND TOTAL MAN MONTHS

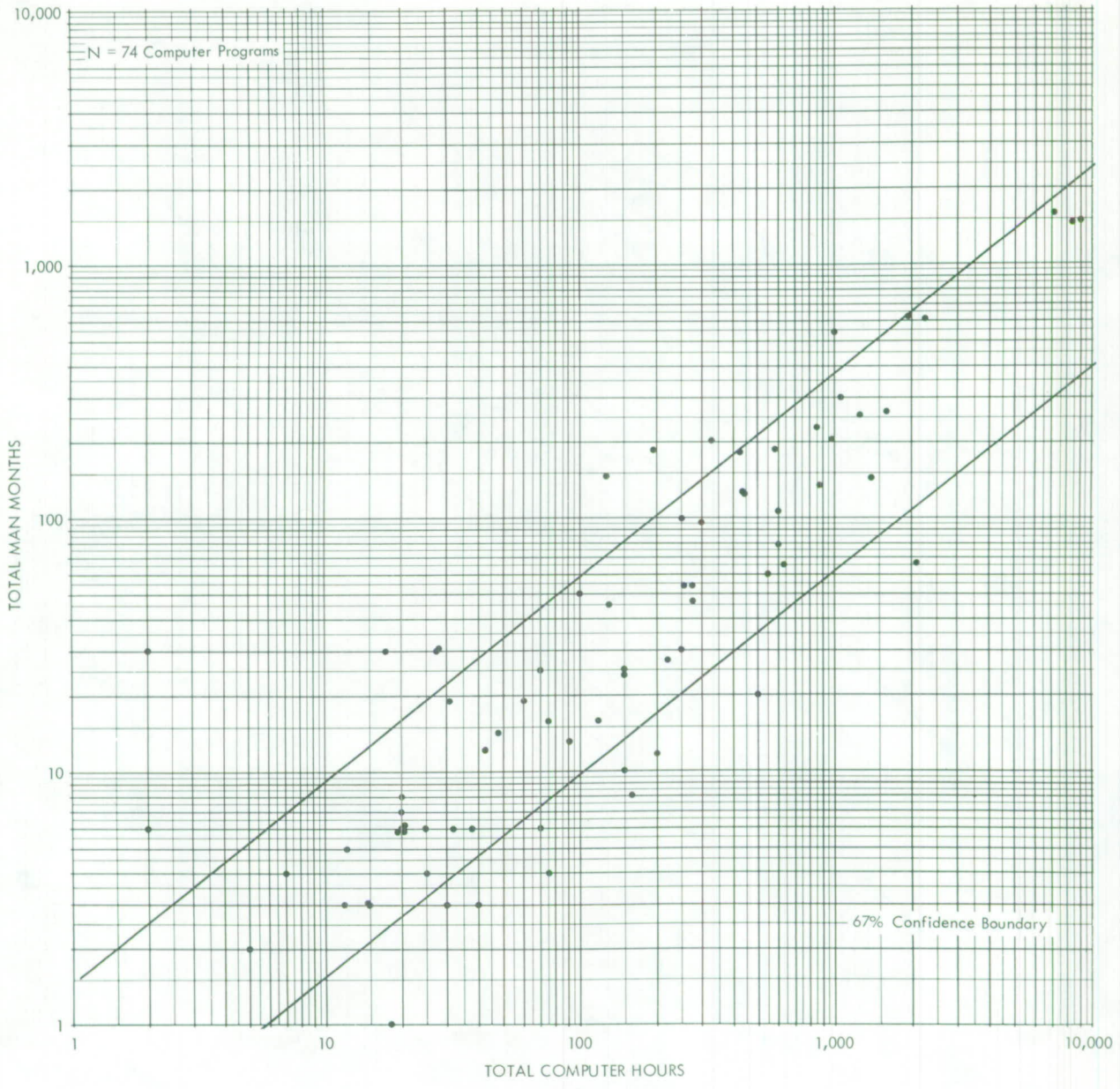


FIGURE 7

RELATIONSHIP BETWEEN TOTAL COMPUTER HOURS AND TOTAL MAN MONTHS

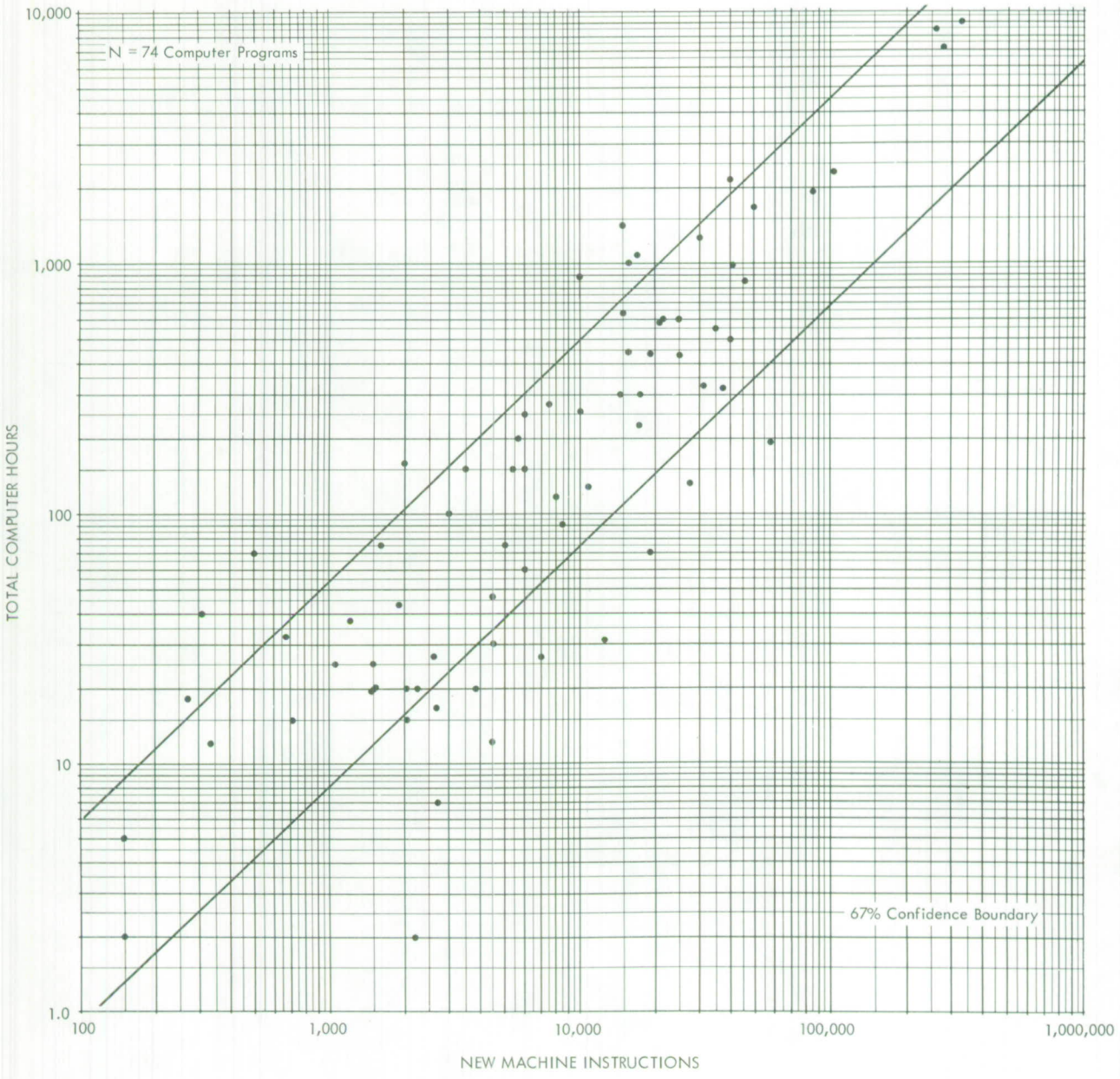


FIGURE 8

RELATIONSHIP BETWEEN NEW MACHINE INSTRUCTIONS AND TOTAL COMPUTER HOURS

3. Identification of Dependent Variables. Based on the correlation analysis results obtained in the previous steps, four primary dependent variables were identified for the remainder of the analysis. These were the same as in the previous study of this series (TM-1447/001/00).

Variable Number	Title
1	Log_{10} Total Man Months
9	Log_{10} Total Computer Hours
20	Log_{10} New Instructions (Machine)
121	Log_{10} Months Elapsed

It is important to note that many other dependent variables could have been added to this list. Each regression analysis essentially represents a separate study effort, however, and time constraints did not permit consideration of additional dependent variables.

Based on the above, all other variables (among the 69) that were not appropriate as predictors, e.g., tended to duplicate the selected dependent variables, were considered for elimination. The result was a list of 40 "independent" variables, shown in Appendix VI. The deletions can be identified by a comparison with Appendix V.

The validities of the more important predictors with the dependent variables, and the Costliness Factor, are provided in Appendix IV.

SECTION XIII

MULTIVARIATE REGRESSION USING INDIVIDUAL PREDICTORS

In what amounted to a repeat of the earlier analysis (TM-1447/001/00), the 4 dependent cost variables were regressed against the 40 independent predictors, and weighted composites of 3 to 8 variables were obtained.

One very significant difference from the previous analysis was that Number of New Instructions, itself an unknown, was no longer being used as an independent variable. The predictors, in each case, either are known or can be estimated with reasonable reliability at the start of program design. While the multiple regression coefficients (an indication of the extent to which the variance of the dependent variable is explained by the predictors) are not directly comparable with those of the previous analysis, due to the absence of Number of New Instructions, the fact that at least one multiple regression coefficient between 0.70 and 0.80 was obtained for all predictions was considered quite encouraging.

For illustrative purposes, several of the equations derived in this manner are shown in Appendix VIII. The others are not included since the techniques that are recommended for prediction involve the use of task indices, which are developed in the following Sections.

SECTION XIV

FACTOR ANALYSIS:

DEVELOPMENT OF THE COSTLINESS FACTOR

1. Background. As noted in Section IV, one of the main deficiencies in the first study, which we sought to remedy in this analysis, was that of aggregation: combining in meaningful and statistically significant ways the substantial number of variables that seemed to affect computer programming resource-costs.

One practical application for an aggregation of this type is the rather common problem of ranking programming jobs as to their "costliness." Managers often want to compare similar tasks to determine the range of costs that can be expected. While dollar estimates are usually available and have the necessary virtue of combining many variables into a single number, the result is often oversimplified, i.e., many managers want to make nonbudgetary comparisons in terms of resource-costs such as man months, number of instructions, etc., rather than dollars. Of course, the programming process is complex enough that a single resource-cost is not a sufficiently meaningful yardstick for most purposes. For this reason, we were seeking a dimensionless aggregate of resource-cost variables that could be used as a measure of the "costliness" of programming jobs.

The correlation analysis clearly indicated that two variables, Pages of Documentation and the Total Number of Trips, were significantly correlated with the major dependent variables, i.e., they tended to reinforce the same variance. The intercorrelation matrix for the primary dependent variables is shown in Table I.

TABLE I

INTERCORRELATION MATRIX FOR PRIMARY DEPENDENT VARIABLES

Var. No.		Log_{10} No. Trips	Log_{10} Pages Doc.	Log_{10} Total Man Months	Log_{10} Total Comp. Hours	Log_{10} New Instr.
10	Log_{10} Number of Trips	1.00	.51	.62	.61	.55
118	Log_{10} Pages of Documentation	.51	1.00	.81	.72	.74
1	Log_{10} Total Man Months	.62	.81	1.00	.86	.87
9	Log_{10} Total Computer Hours	.61	.72	.86	1.00	.86
20	Log_{10} New Instructions (Machine)	.55	.74	.87	.86	1.00

This interdependency suggested the application of factor analysis techniques, i.e., to the definition of a single underlying factor in terms of which the variance contributed by the components above could be represented.

Although no distinct pattern of intercorrelations was apparent among the independent variables, we hoped that factor analysis would also reveal underlying relationships that would permit the grouping of potential predictors.

2. Factor Analysis Results. The first 25 principal factors derived from the analysis of 69 dependent and independent variables were studied before and after rotation (a technique to sharpen the distinctions between significantly related groupings). The relation among the five aspects of resource-cost mentioned previously was delineated in the first factor, which is shown in Appendix VII.

Both the 69 dependent and independent variables, and the 40 independent variables, were analyzed with factor analysis techniques for groupings among the independent variables. The results were rather disappointing. No distinct and interpretable combinations of independent variables were selected. However, a more extensive factor analysis and the application of more sophisticated techniques (12) may well have revealed acceptable predictor factors. Since time was limited, a different approach to grouping predictors was successful and is discussed in the next Section. We intend to consider factor analysis again when a larger data base is available.

3. Development of the Costliness Factor. To obtain proper weighting, the Costliness Factor was regressed against its five components. The resultant regression coefficients (14) were used to form the factor, and are shown in Appendix VII. The validities of the component variables with the Costliness Factor, i.e., a measure of the degree to which their variance is represented, is indicated by Table II:

TABLE II

VALIDITY OF THE COMPONENTS WITH THE COSTLINESS FACTOR

Var. No.		Validity of Weighted Components with the Costliness Factor, Variable 154
10	Log_{10} Number of Trips	.73
118	Log_{10} Pages of Documentation	.80
1	Log_{10} Total Man Months	.91
9	Log_{10} Total Computer Hours	.88
20	Log_{10} New Instructions (Machine)	.88

The multiple correlation coefficient between all of the weighted components and the Costliness Factor is 0.96 out of a maximum of 1.00.

The frequency distribution of "costliness" for the 74 data points is shown in Figure 9, based on a mean of 100 and a standard deviation of 20.

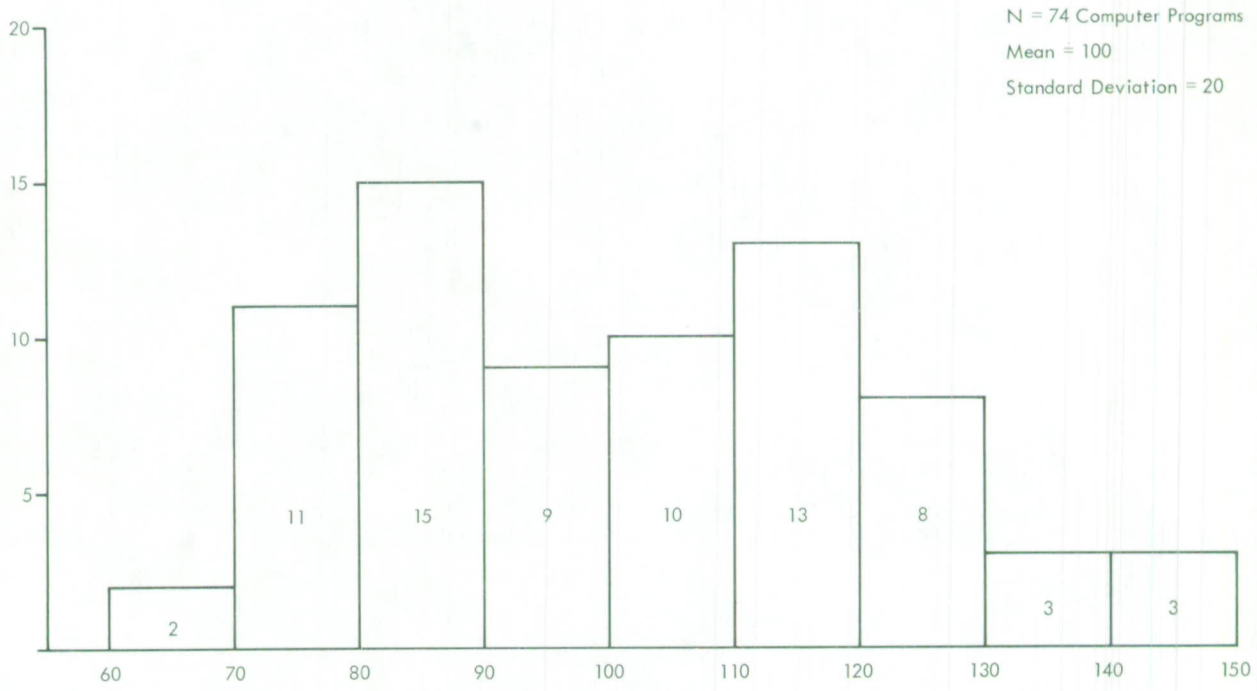


FIGURE 9

FREQUENCY DISTRIBUTION OF THE COSTLINESS FACTOR

Note that, considering the lack of selection criteria for the data base, the distribution is rather well proportioned, i.e., tends to unimodality and symmetry.

(Page 50 Blank)

SECTION XV

THE DEVELOPMENT OF TASK INDICES

1. Summary of Prior Analytical Steps. The development of indices that describe the programming task is presented in this Section, and we consider these groupings as the major analytical finding of the analysis. However, the delineation of meaningful aggregates of variables was dependent on the availability of a working set of predictor variables, i.e., one from which spuriousity and redundancy have been removed. This required the preparatory analytical steps discussed in Sections IX through XIV.

First, the questionnaires were studied for accuracy and completeness. Suspicious (or nonexistent) responses were discussed with the programmer to resolve misinterpretations whenever possible. In about one percent of the cases, values were estimated by the Project staff. Since data were not evenly distributed for many variables, i.e., there were many smaller values close together, and fewer large values rather widely separated, logarathmic transformations were applied in many instances to achieve a more continuous and more normal array.

Next, scatterplot and correlation analysis were used to study the relationships among the variables. Those which had low validities, i.e., were not strongly related to the dependent cost variables, were considered for elimination, as were those predictors that tended to duplicate each other's contribution to prediction. Factor analysis was then applied in an attempt to obtain groupings of variables. A nondimensional aggregate of related aspects of resource-costs was defined, and termed "Costliness." However, the factor analysis procedures failed to delineate any distinct and interpretable groupings of predictors.

As a result, we tried another approach to obtain combinations of independent variables. This involved what might be thought of as the opposite of factor analysis: the grouping of variables that contributed intuitively similar but statistically different ingredients to the prediction. Initially, the 40 independent variables were divided into 5 intuitively attractive groupings that corresponded to different influences on computer programming costs. The components in each group with relatively high intercorrelations and low validities with the dependent variables were then weeded out in favor of those predictors with the lowest intercorrelations and the highest validities. The development of each of the indices is considered separately in this context.

2. The Uniqueness Index. Ten of the 40 independent variables shared the characteristic of being rather uncommon, yet having a substantial impact on program cost when present. These variables were considered for a measure of the uniqueness of the task. The initial correlation and validity relationships for the 10 variables evaluated are shown in Table III. Note that only the validities with the Costliness Factor are shown; however, validities with each of the dependent variables, i.e., Man Months, Computer Hours, Number of New Instructions, and Months Elapsed, were also considered before deletions were made.

The intent of the winnowing process was to obtain a few predictors with significant cost validities, i.e., predictive power and relatively low intercorrelations, i.e., redundancy. The resulting four components are shown in Table IV. Note that all of the predictors in Table IV are two-valued, i.e., always either one or zero, so that, when uniqueness as defined is not present, the index adds zero to the overall prediction.

To weight the predictors properly in terms of their contribution to prediction, each of the four dependent variables, and the Costliness Factor, were separately regressed against the four components of the Uniqueness Index. The raw regression coefficients for each variable were averaged, and rescaled, to obtain the following weights[8]:

1.2 x Innovation in System

1.0 x Stringent Timing Requirements

1.7 x First Programming Effort on Computer

1.3 x Program Developed at More Than One Location

The Uniqueness Index was defined as the sum of the weighted variables as shown and is identified for this analysis as variable 149. The maximum value for this index is 5.2; the minimum zero, and its validity with the Costliness Factor is 0.52. (Note that the validity of the weighted and summed ingredients of the index is considerably greater than for any of the components taken separately.)

The distribution of the Uniqueness Index for the 74 data points is shown in Figure 10, based on a mean of 100 and a standard deviation of 20.

---

[8]The details of the regression and weighting process for this index are provided in Appendix IX.

TABLE III

CORRELATION MATRIX OF VARIABLES CONSIDERED FOR THE UNIQUENESS INDEX

Var. No.		Innovat. in Syst.	Insuf. Mem.	Insuf. I/O	String. Timing	First Eff. on Comp.	Log_{10} ADP Comp. Dev.	Comp. Oper. by Other Agency	Comp. at Other Site	Ops. Comp. Diff.	Prog. Dev. at More Than One Location	Validity With Costliness Factor	Comments
12	Innovation in System	1.00	- .05	.12	.09	.22	.14	- .06	- .47	.00	- .11	.29	
50	Insufficient Memory Capacity	- .05	1.00	.41	.38	.06	.27	- .07	.20	.05	.22	.27	Deleted in favor of variable 52.
51	Insufficient I/O Capability	.12	.41	1.00	.32	- .12	- .10	.27	.02	- .29	.20	.18	Deleted in favor of variable 52.
52	Stringent Timing Requirement	.09	.38	.32	1.00	- .13	.12	.09	.15	- .02	.18	.24	
64	Was This First Programming Effort on the Computer	.22	.06	- .12	- .13	1.00	.43	- .20	- .18	.22	- .04	.30	
66	Log_{10} Number ADP Components Developed Concurrently with Program	.14	.27	- .10	.12	.43	1.00	- .08	.00	.09	.03	.28	Deleted in favor of variable 64.
101	Was Computer Operated by Agency Other Than Program Developer	- .06	- .07	.27	.09	- .20	- .08	1.00	.14	- .39	.21	- .02	Deleted due to low validity.
104	Was Computer Operated at a Site Other Than Operational Location	- .47	.20	.02	.15	- .18	.00	.14	1.00	.22	.46	.15	Deleted in favor of variable 106.
105	Is Operational Computer Different Than Development Computer	.00	.05	- .29	- .02	.22	.09	- .39	.22	1.00	.13	.11	Deleted in favor of variable 106.
106	Did Program Development Take Place at More Than One Location	- .11	.22	.20	.18	- .04	.03	.21	.46	.13	1.00	.27	

TABLE IV

CORRELATION MATRIX OF VARIABLES SELECTED FOR THE UNIQUENESS INDEX

Var. No.		Innov.	Stringent Timing	First Effort	Ops. Comp. Diff.	Prog. Dev. at More Than 1 Loc.	Validity with Costl. Factor
12	Innovation in System	1.00	.09	.22	.00	- .11	.29
52	Stringent Timing Requirements	.09	1.00	- .13	- .02	.18	.24
64	First Programming Effort on Computer	.22	- .13	1.00	.22	- .04	.30
106	Program Developed at More Than One Location	- .11	.18	- .04	.13	1.00	.27

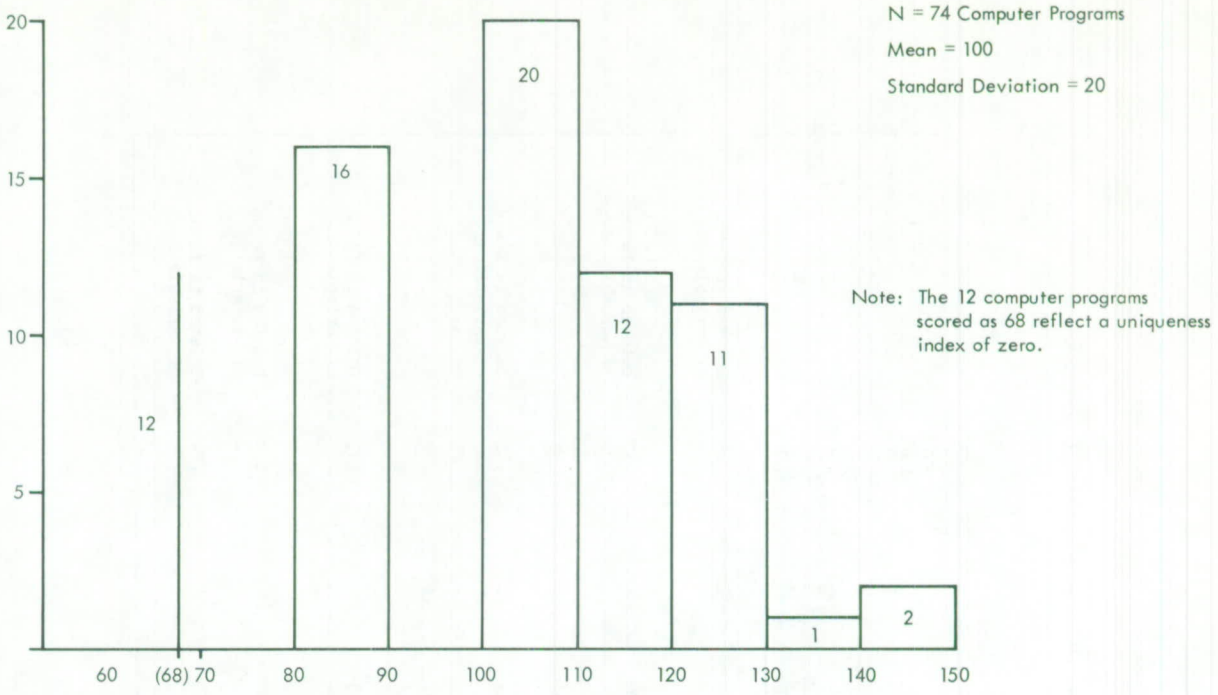


FIGURE 10

FREQUENCY DISTRIBUTION OF THE UNIQUENESS INDEX

3. The Job Difficulty Index. Eight of the 40 predictor variables were intuitively grouped into a measure of the difficulty of the programming job. The intercorrelations and illustrative validities with the Costliness Factor for these 8 variables is shown in Table V.

The intercorrelations and validities for the predictor variables selected as components of the Job Difficulty Index are as follows:

TABLE VI

CORRELATION MATRIX OF THE VARIABLES SELECTED FOR THE JOB DIFFICULTY INDEX

Var. No.		Log_{10} No. Subprog.	Log_{10} No. Classes in D.B.	Validity With Costliness Factor
30	Log_{10} Number of Subprograms	1.00	.13	.57
32	Log_{10} Number of Classes in Data Base	.13	1.00	.22

TABLE V

CORRELATION MATRIX OF THE VARIABLES CONSIDERED FOR THE JOB DIFFICULTY INDEX

Var. No.		Log_{10} No. Users	Log_{10} No. ADP Ctrs.	Log_{10} No. Subprog.	Log_{10} No. Words D. Base	Log_{10} No. Classes D. Base	Log_{10} No. Words Not in D. Base	Log_{10} No. Input Msg. Types	Log_{10} No. Output Types	Validity With Costliness Factor	Comments
15	Log_{10} Number Users Supplying Inputs and Requesting Outputs	1.00	- .04	.05	.18	.32	.05	.16	.09	.14	Deleted in favor of variable 32.
16	Log_{10} Number ADP Centers in System	- .04	1.00	.09	- .28	- .28	- .10	- .07	- .04	- .17	Deleted in favor of variable 32.
30	Log_{10} Number of Subprograms	.05	.09	1.00	.30	.13	- .28	.42	.39	.57	
31	Log_{10} Number of Words in Data Base	.18	- .28	.30	1.00	.55	.48	.11	.21	.41	Deleted in favor of variable 32.
32	Log_{10} Number of Classes in Data Base	.32	- .28	.13	.55	1.00	.42	.21	.33	.22	
33	Log_{10} Number of Words in Tables and Constants Not in Data Base	.05	- .10	.28	.48	.42	1.00	.43	.48	.27	Deleted in favor of variable 32.
34	Log_{10} Number Input Message	.16	- .07	.42	.11	.21	.43	1.00	.80	.29	Deleted in favor of variable 30.
35	Log_{10} Number Output Message Types	.09	- .04	.39	.21	.33	.48	.80	1.00	.24	Deleted in favor of variable 30.

As before, the four dependent variables, and the Costliness Factor were regressed against the two components of the Job Difficulty Index, and averaged and rescaled coefficients used to obtain proper weighting. The final form of the Index was defined as the sum of the weighted variables as shown:

$$5.0 \times \text{Log}_{10} \text{ Number of Subprograms}$$
$$1.0 \times \text{Log}_{10} \text{ Number of Classes in Data Base}$$

Note that, due to the logarithmic transformations, the addition of weighted variables really represents the multiplication of power functions of the raw data. The Job Difficulty Index has a lower limit of zero, and no upper limit, and is identified in this analysis as variable 150. The reader should note that any program in its entirety, is considered to represent at least one subprogram and one class of data. So, in computing this index, the recorded values of the two raw components are always increased by one. Since the $\log_{10}$ of one is zero, the lower limit for the index occurs when the recorded values of the raw components are zero and the entire program is considered to be one subprogram and one class of data.

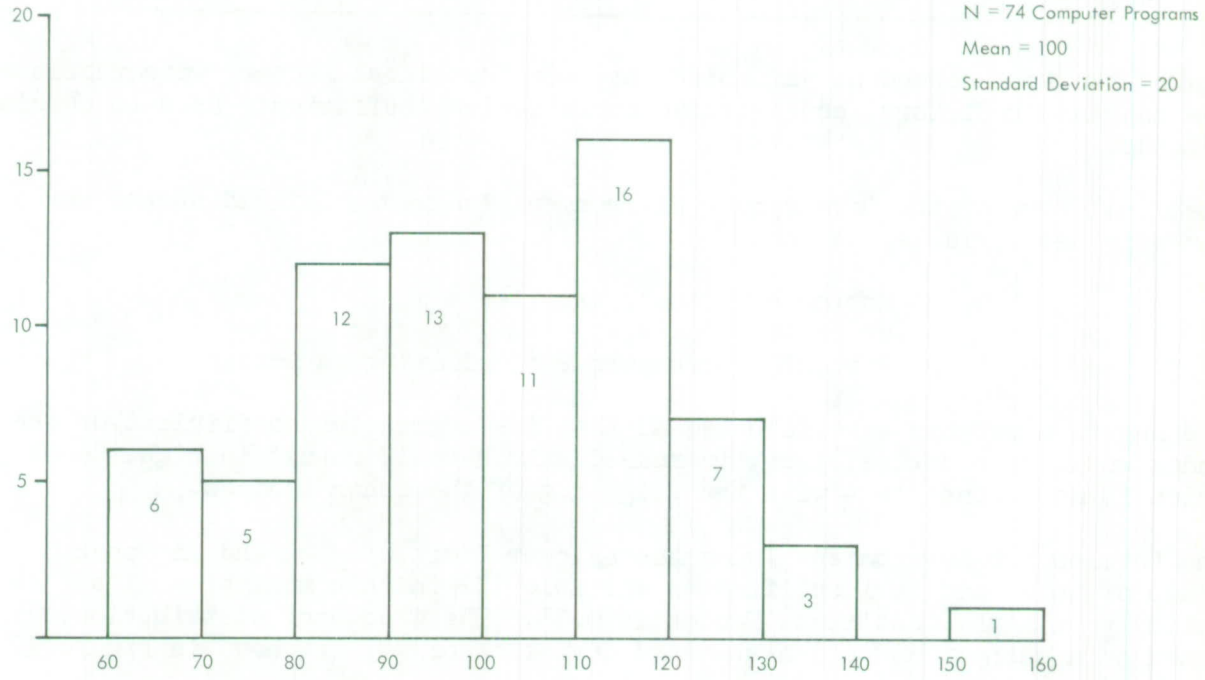


FIGURE 11

FREQUENCY DISTRIBUTION OF THE JOB DIFFICULTY INDEX

4. Development Environment Index. Seven of the 40 predictor variables were intuitively grouped into a measure of the development environment. The intercorrelations and illustrative validities of these 7 variables are shown in Table VII.

The intercorrelations and validities for the predictors selected as components of the Development Environment Index are as follows:

TABLE VIII

CORRELATION MATRIX OF THE VARIABLES

SELECTED FOR THE DEVELOPMENT ENVIRONMENT INDEX

Var. No.		Est. Cust. Exper.	% Prog. In Des.	Validity With Costliness Factor
100	Estimate Customer Experience	1.00	.04	- .21
143	Percent Programmers Participating in Design	.04	1.00	- .49

Again, the four dependent variables, and the Costliness Factor, were regressed against the predictors, and averaged and rescaled coefficients used to obtain weights.

The final form of the Development Environment Index was defined as the sum of the weighted variables as shown:

-1.0 x Estimate Customer Experience

-3.4 x Percent Programmers Participating in Design

The negative weights were selected so that the regression coefficient of the Index would be positive, i.e., increased values would result in higher costs, which is more consistent with the weighting of the other indices.

The Development Environment Index has a lower limit of -1.0 and an upper limit of -6.4, and is identified as variable 151 in this analysis. Its validity with the Costliness Factor is 0.53. The frequency distribution, based on a mean of 100 and a standard deviation of 20, is shown in Figure 12.

TABLE VII

CORRELATION MATRIX OF THE VARIABLES CONSIDERED FOR THE DEVELOPMENT ENVIRONMENT INDEX

Var. No.		How Well Plans Known	Log_{10} Avg. Turn. Time	No. Agen. Concurr.	Est. Cust. Experience	Open/Closed Shop	Time-Sharing	Percent of Prog. Part. in Design	Validity With Costliness Factor	Comments
14	How Well Were Plans Known and Documented	1.00	- .22	.14	.17	.16	- .17	.05	- .04	Deleted due to low validity.
65	Log_{10} Average Turnaround Time	- .22	1.00	- .28	- .20	.30	- .28	- .28	.17	Deleted due to redundancy.
99	Number of Agencies Concurring in Design	.14	- .28	1.00	.08	.19	- .13	- .02	.13	Deleted due to low validity.
100	Estimate Customer Experience	.17	- .20	.08	1.00	- .04	- .06	.04	- .21	
102	Open/Closed Shop	.16	.30	.19	- .04	1.00	- .16	.08	.03	Deleted due to low validity.
103	Time-Sharing	- .17	- .28	- .13	- .06	- .16	1.00	.19	- .18	Deleted due to interrelation with variables 14 and 65.
143	Percent of Programmers Participating in Design	.05	- .28	- .02	.04	.08	.19	1.00	- .49	

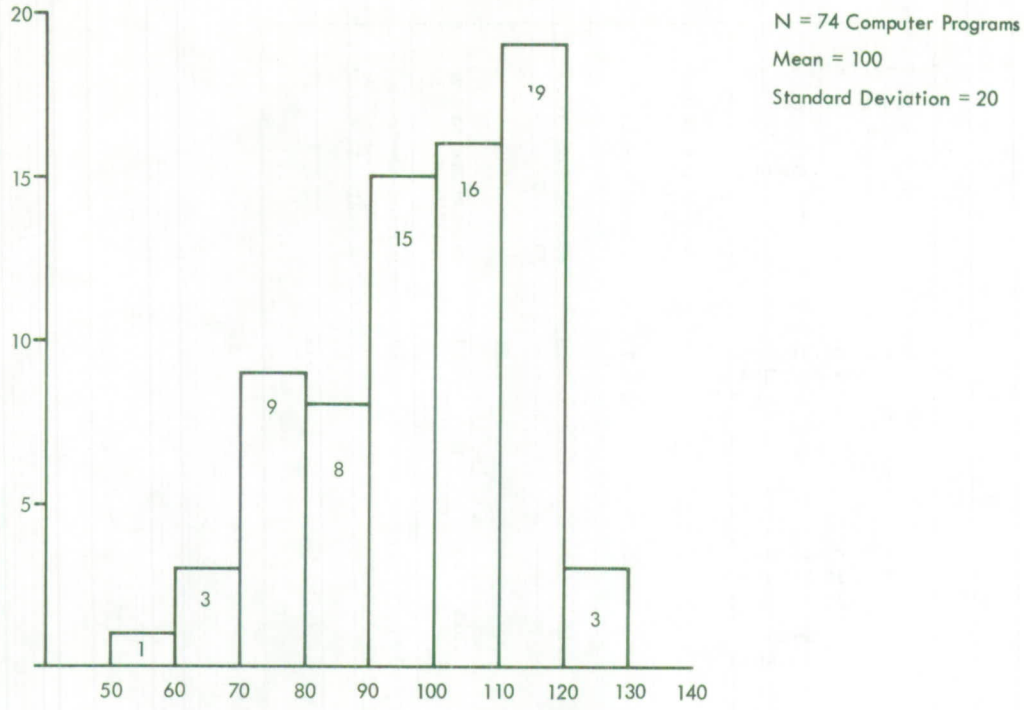


FIGURE 12

FREQUENCY DISTRIBUTION OF THE DEVELOPMENT ENVIRONMENT INDEX

5. Job Type Index. Nine of the 40 predictor variables were intuitively grouped into a measure of the types of data-handling that were characteristic of the programming job. The intercorrelations and illustrative validities for these 9 variables are shown in Table IX.

The intercorrelations and validities for the variables selected as components of the Job Type Index are as follows:

TABLE X

CORRELATION MATRIX OF THE VARIABLES SELECTED FOR THE JOB TYPE INDEX

Var. No.		% Clerical	% Trans. & Ref.	% Gener.	Valid. With Costl. Factor
37	Percent Clerical Instructions	1.00	.17	- .14	- .23
46	Percent Transformation and Reformatting Function	.17	1.00	.08	- .13
47	Percent Generation Function	- .14	.08	1.00	.29

TABLE IX

CORRELATION MATRIX OF THE VARIABLES CONSIDERED FOR THE JOB TYPE INDEX

Var. No.		Percent Clerical	Percent Math	Percent I/O	Percent Control	Percent Self-Check	Percent Stor. & Ret.	Percent Acq. & Disp.	Percent Transf. & Reform.	Percent Generation	Validity With Costliness Factor	Comments
37	Percent Clerical Instructions	1.00	- .30	- .46	- .44	- .09	.53	- .38	.17	- .14	- .23	
38	Percent Math Instructions	- .30	1.00	- .23	- .19	- .01	- .22	.25	- .21	.24	.22	Deleted in favor of variable 37.
39	Percent I/O Instructions	- .46	- .23	1.00	- .26	- .20	- .21	.23	.21	.06	.02	Deleted due to low validity.
40	Percent Logical Control Instructions	- .44	- .19	- .26	1.00	- .02	- .25	.00	- .18	- .08	.08	Deleted due to low validity.
41	Percent Self-Check Fix Instructions	- .09	- .01	- .20	- .02	1.00	.07	- .04	- .20	.05	.04	Deleted due to low validity.
42	Percent Information Storage and Retrieval	.53	- .22	- .21	- .25	.07	1.00	- .34	- .13	- .25	- .16	Deleted in favor of variable 37.
43	Percent Data Acquisition and Display	- .38	.25	.23	.00	- .04	- .34	1.00	- .35	- .28	.08	Deleted due to low validity.
46	Percent Transformation-Reformatting	.17	- .21	.21	- .18	- .20	- .13	- .35	1.00	.08	- .13	
47	Percent Generation	- .14	.24	.06	- .08	.05	- .25	- .28	.08	1.00	.29	

The 4 dependent variables, and the Costliness Factor, were again regressed against the 3 components of the Index, and averaged and rescaled coefficients used to obtain weights. The final form of the Job Type Index was defined as the sum of the weighted variables as shown:

- 1.0 x Percent Clerical Instructions

- 1.3 x Percent Transformation-Reformatting Function

4.4 x Percent Generation Function

The Job Type Index has a lower limit of - 1.3 and an upper limit of 4.4, and is identified in this analysis as variable 152. Its validity with the Costliness Factor is 0.45. The frequency distribution, based on a mean of 100 and a standard deviation of 20 is shown in Figure 13.

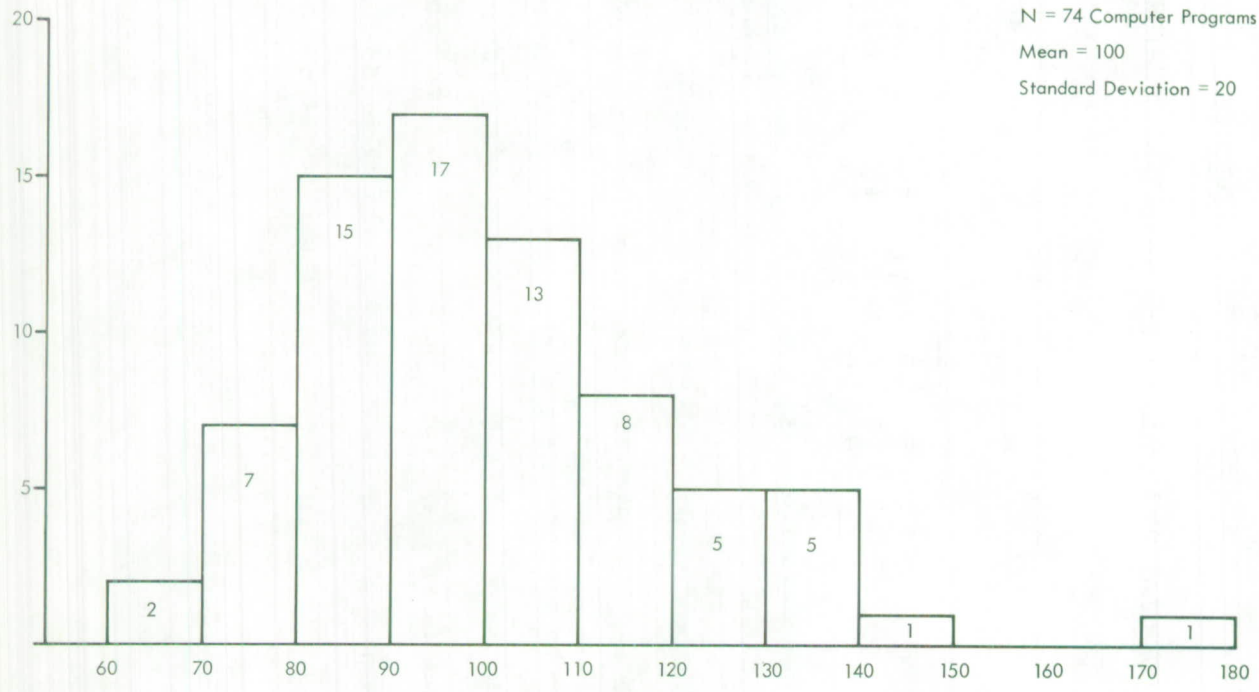


FIGURE 13

FREQUENCY DISTRIBUTION OF THE JOB TYPE INDEX

6. Staffing Index. Several attempts were made during the analysis to define a useful measure of the experience and capability of the programming staff. None of these attempts demonstrated statistical significance.

We intend to pursue this matter further, especially since a measure of staff capability has such a high intuitive appeal as a predictor of programming costs.

SECTION XVI

AN EVALUATION OF THE TASK INDICES

As we have noted previously, the main appeal of the task indices is not so much their statistical significance as their interpretability. After the winnowing process discussed in Section XV and the resulting eleven predictor variables could have been directly combined into regression equations along the lines of those discussed in Section XIII, i.e., without grouping. While the presence of more than four or five predictors in a single linear regression equation is rather unusual, and could lead to undesirable weighting, the statistical power would likely not be a great deal different than if the task indices had been used. In other words, the relationships between the eleven independent and the dependent variables are what underly the prediction, whether the independents are weighed and grouped into indices or not. In this sense, the indices reflect our desire to assure that a more comprehensive set of predictors are represented in the equation, and suitably weighted, rather than limit the number of predictors to four or five (and run the risk of oversimplification).

Moreover, the indices facilitate the comparison of programming jobs, e.g., whether one is more unique, or difficult than another, and the planning that managers do to reduce costs, e.g., an increase in programmer participation in program design to lessen the contribution to cost of the Development Environment Index.

It is clear, however, that the four indices developed in this analysis are rather rudimentary. While each of the eleven components tend to represent the contribution of many other variables, i.e., each of the eleven is related to others that are not components of an index, there are other aspects of the programming process that are omitted. We intend to refine and augment the present indices in subsequent analyses of this series.

SECTION XVII

THE ESTIMATION OF DEPENDENT COST VARIABLES USING TASK INDICES

1. General Approach. We are now ready to translate the results presented in Sections IX through XVI into a management tool for estimating the primary resource-cost variables, i.e., Man Months, Computer Hours, Months Elapsed, and New Instructions, and the Costliness Factor. To obtain regression estimation equations, each of these dependent variables was separately regressed against the task indices developed in Section XVI. The components of each index are summarized in Table X.

The resultant regression equations are shown in Table XI[9]. The technical details of the regressions are provided in Appendix X.

2. Development of the Stanine Plots. It is customary in statistical estimation to construct confidence bands that straddle the prediction function and to establish the limits within which a certain proportion of all observed values will fall, for equally-sized samples[10]. Although the confidence-interval approach provides a useful safeguard against unwarranted reliance on computed estimates, single bands straddling the prediction line are too inflexible for estimating computer programming costs. The reason is that the values near each extreme of the band have a much lower probability of occurring than those near the center, i.e., observations tend to be normally distributed around each estimate.

---

[9] The computed prediction function that minimized the least-squares variance of the observed values tended to provide overestimates of smaller programs and underestimates of larger programs. (This was due in part to the fact that the constant in the regression equation tends to inflate smaller estimates, and in part to the nature of the least-squares procedure.) For this reason, the function was rotated slightly to equalize the variance of the estimated and observed points. The effect of this adjustment was to distribute the positive and negative estimation variances more equally along the entire range of interest. The loss in predictive efficiency was less than two percent.

[10] In principle, this means that the 0.95 confidence band will encompass 95 percent of all observations based on a given estimate, for equally-sized samples.

TABLE XI

COMPONENTS OF THE TASK INDICES

Index	Weight	Component Variable	Variable Number
Uniqueness	1.2	Innovation in System	12
	1.0	Stringent Timing Requirements	52
	1.7	First Programming Job on Computer	64
	1.3	Program Developed at More Than One Location	106
Job Difficulty	5.0	Log_{10} Number of Subprograms	30
	1.0	Log_{10} Number of Classes in Data Base	32
Development Environment	-1.0	Estimate Customer Experience	100
	-3.4	Percent Programmers Participating in Design	143
Job Type	-1.0	Percent Clerical Instructions	37
	-1.3	Percent Transformation and Reformatting Function	46
	4.4	Percent Generation Function	47

NOTE: In the computation for the Job Difficulty Index, the values of each raw component should be increased by one, since any programming task, i.e., data point in its entirety, is considered to represent a difficulty equal to at least one subprogram and one class of data.

TABLE XII

COST ESTIMATION EQUATIONS

Log_{10} Total Man Months	= 0.23 Uniqueness Index + 0.08 Job Difficulty Index + 0.21 Development Environment Index + 0.36 Job Type Index + 1.56
Log_{10} Total Computer Hours	= 0.28 Uniqueness Index + 0.12 Job Difficulty Index + 0.19 Development Environment Index + 0.25 Job Type Index + 1.69
Log_{10} Months Elapsed	= 0.11 Uniqueness Index + 0.06 Job Difficulty Index + 0.05 Development Environment Index + 0.12 Job Type Index + 0.60
Log_{10} New Instructions (Machine)	= 0.29 Uniqueness Index + 0.08 Job Difficulty Index + 0.13 Development Environment Index + 0.32 Job Type Index + 3.31
Costliness Factor	=18.93 Uniqueness Index + 3.58 Job Difficulty Index + 5.89 Development Environment Index + 10.37 Job Type Index + 66.96

(Page 68 Blank)

To provide the user of the cost estimation plots with a more flexible means of taking the probability distribution of programming tasks into account and blending in his own experience, a novel confidence-band rationale based on an application of the Stanine scoring approach was used(20).

Nine confidence bands were constructed, with the center band straddling the prediction line and four bands on each side. The two outer bands (1 and 9) are open-ended. All the other bands (2 through 8) are of equal width. The probability of observed values falling into each of 9 Stanine confidence bands is as follows:

Stanine Band	Probability of Occurrence
1	.04
2	.07
3	.12
4	.17
5	.20
6	.17
7	.12
8	.07
9	.04

Note that the middle, or "5" band, represents the most probable or expected value. As the bands differ from the "5" band, they contain values that are less and less likely to occur. The application of these bands to estimation is illustrated best with an example. Consider Figure 14, where estimated versus actual values for the 74 points in the data base are plotted for the Costliness Factor. The middle of the center, or "5" band, is equivalent to the expected resource-cost computed from the regression equation. The limits of the "5" band define the variation around the computed estimate for the average case. The "6" through "9" bands define progressively higher variations from the computed estimate; the "4" through "1" bands define progressively lower variations. Note that the points tend to be clustered along the prediction line, which would pass through the middle of the "5" band. In fact, this tendency exceeds the theoretical proportion. For instance, nearly 27 percent of the points fall into the "5" band, more than the 20 percent expected. This effect is due to the nature of the plot, which depicts after-the-fact relationships. If estimates were made with new data, and not from the 74-point base used to derive the equations, we would expect the distribution of observations around the estimates to be closer to the theoretical values.

(Page 70 Blank)

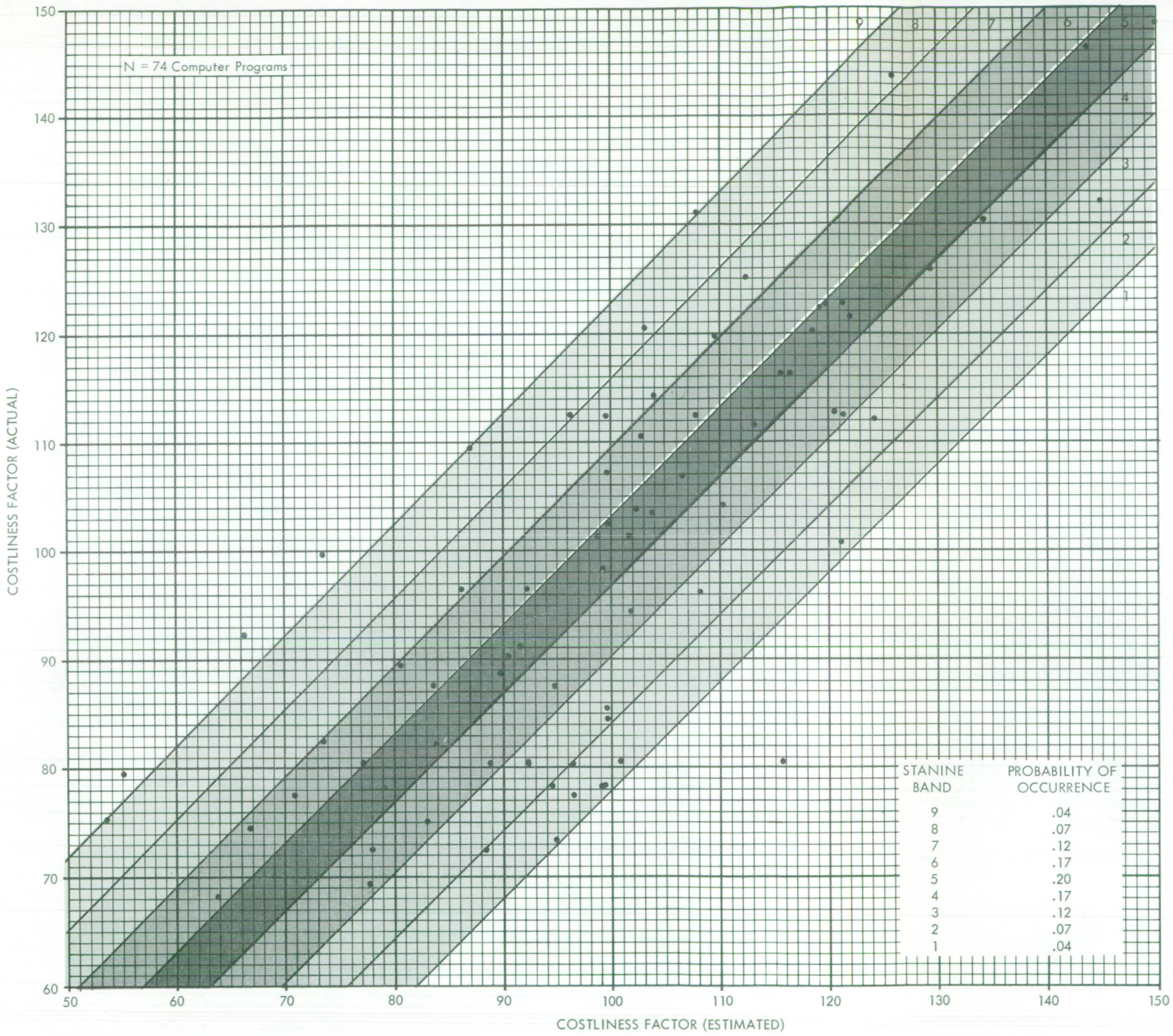


FIGURE 14

STANINE PLOT FOR COSTLINESS

(Page 72 Blank)

The center, or "5" band, is equivalent to a 0.20 confidence estimation for the Costliness Factor. The "4," "5," and "6" bands together are equivalent to a 0.54 confidence estimation (.17 + .20 + .17). Higher-valued confidence bands can be constructed simply by summing the appropriate probabilities of occurrence.

3. Application of the Plots. Suppose that a manager felt that a proposed programming task, when compared to the spectrum of possible programs, represented an "average" type of job, i.e., a costliness of 100. What is the probability that costliness scores of, say, 130 could result? By following the 130 line horizontally to the vertical 100 line, we see that the intersection falls in the high "9" band. If the manager's intuitive evaluation of the task as "average" was correct, a costliness score of 130 is quite unusual; i.e., it would occur less than 4 percent of the time. In retrospect, a program product that scores 130 in costliness is quite unusual from any point of view (less than four percent of all jobs will score 130 or higher), regardless of the manager's a priori evaluation. The Stanine plot for costliness thus provides the manager with a tool for establishing the limits of variation to be expected around an estimate for various degrees of confidence.

The Stanine plots can be useful in other ways when one of the other dependent variables is involved (see Figures 15 through 18 which follow). Consider Figure 15, for example, which depicts estimated versus actual values for man months. Suppose that, based on the appropriate equation in Table XII, a user computes an estimate of 100 man months for a particular programming job (i.e., a value of 2.0 in logarithms). How much of a variance around this estimate should be expected? Several options are available:

a. An "average" estimate for man months, based on a computed value of 100, would involve confidence-band limits of 73 and 130[11]. The probability of values occurring in this range is at least 0.20.

b. Based solely on his intuitive appraisal of the task, the components of the prediction equation, his experience, or any other factors pertinent to the problem, the user can consider the programming job as equivalent to a higher or lower confidence band, and read the upper and lower estimates directly. A score of "6," for example, would imply a program slightly above average, and would yield confidence band limits of 130 and 230 man months. The probability of values occurring in this range is at least 0.17.

---

[11] The limits are not symmetrical around the estimate due to the use of logarithmically-transformed variables.

(Page 74 Blank)

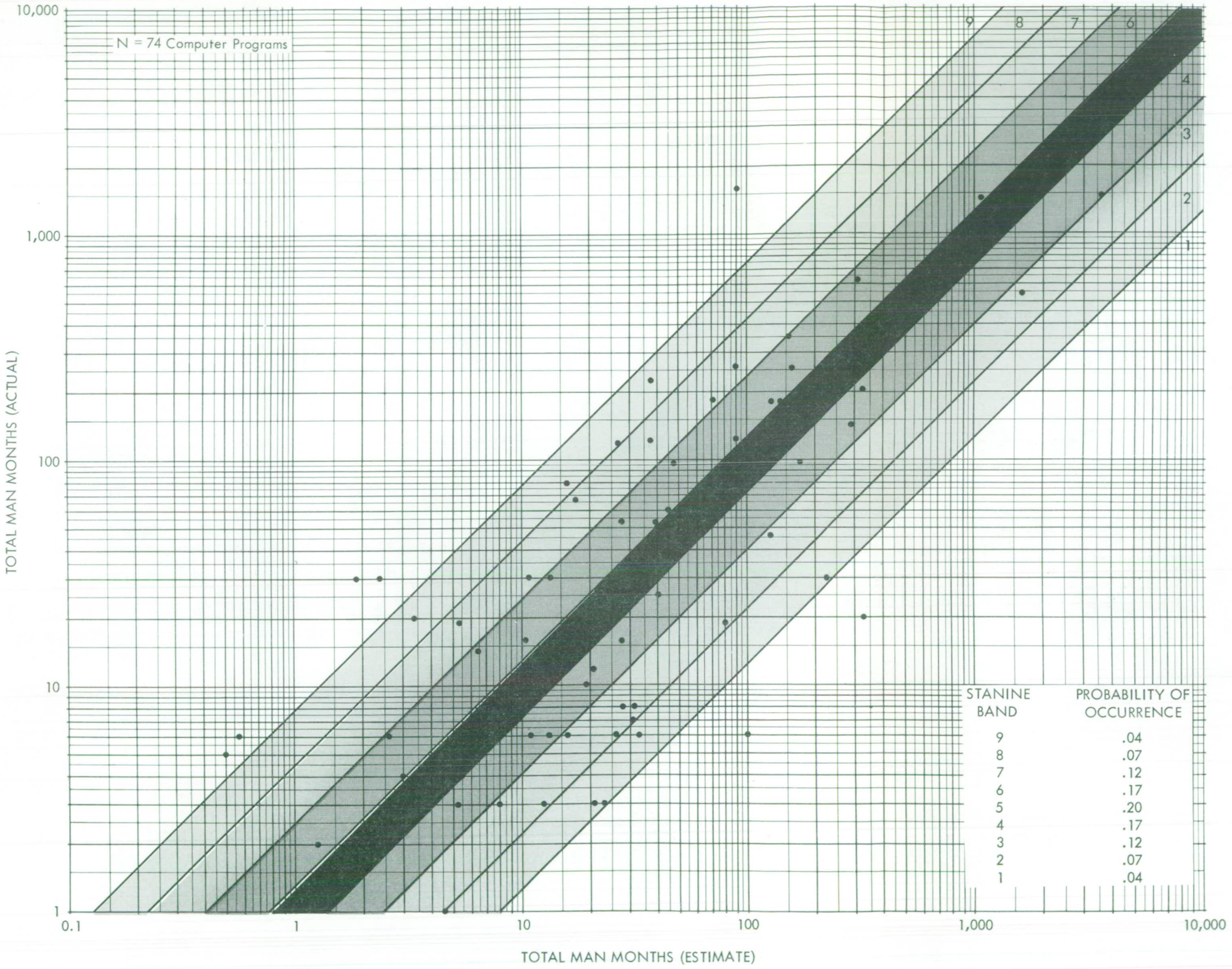


FIGURE 15

STANINE PLOT FOR MAN MONTHS

(Page 76 Blank)

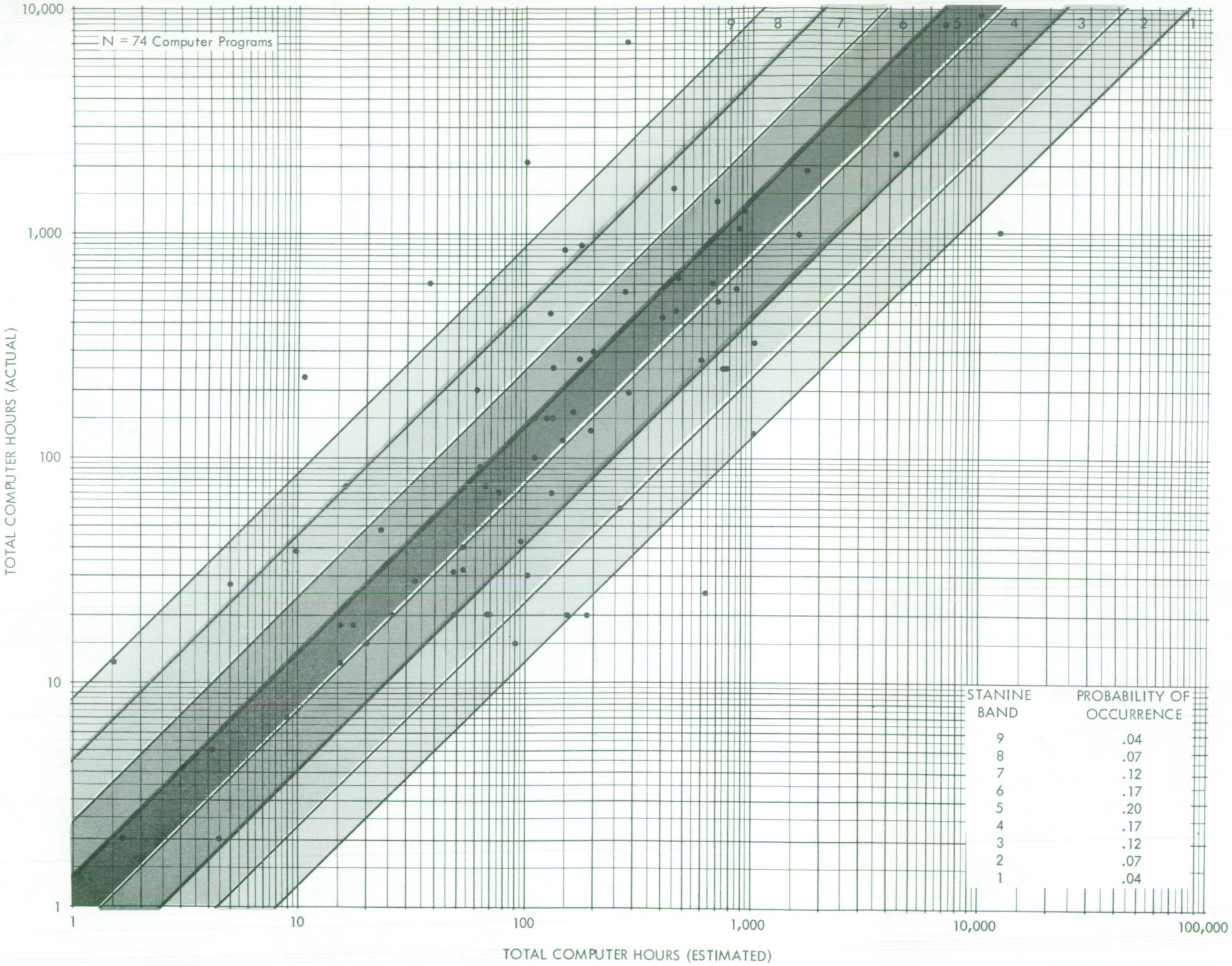


FIGURE 16

STANINE PLOT FOR COMPUTER HOURS

(Page 78 Blank)

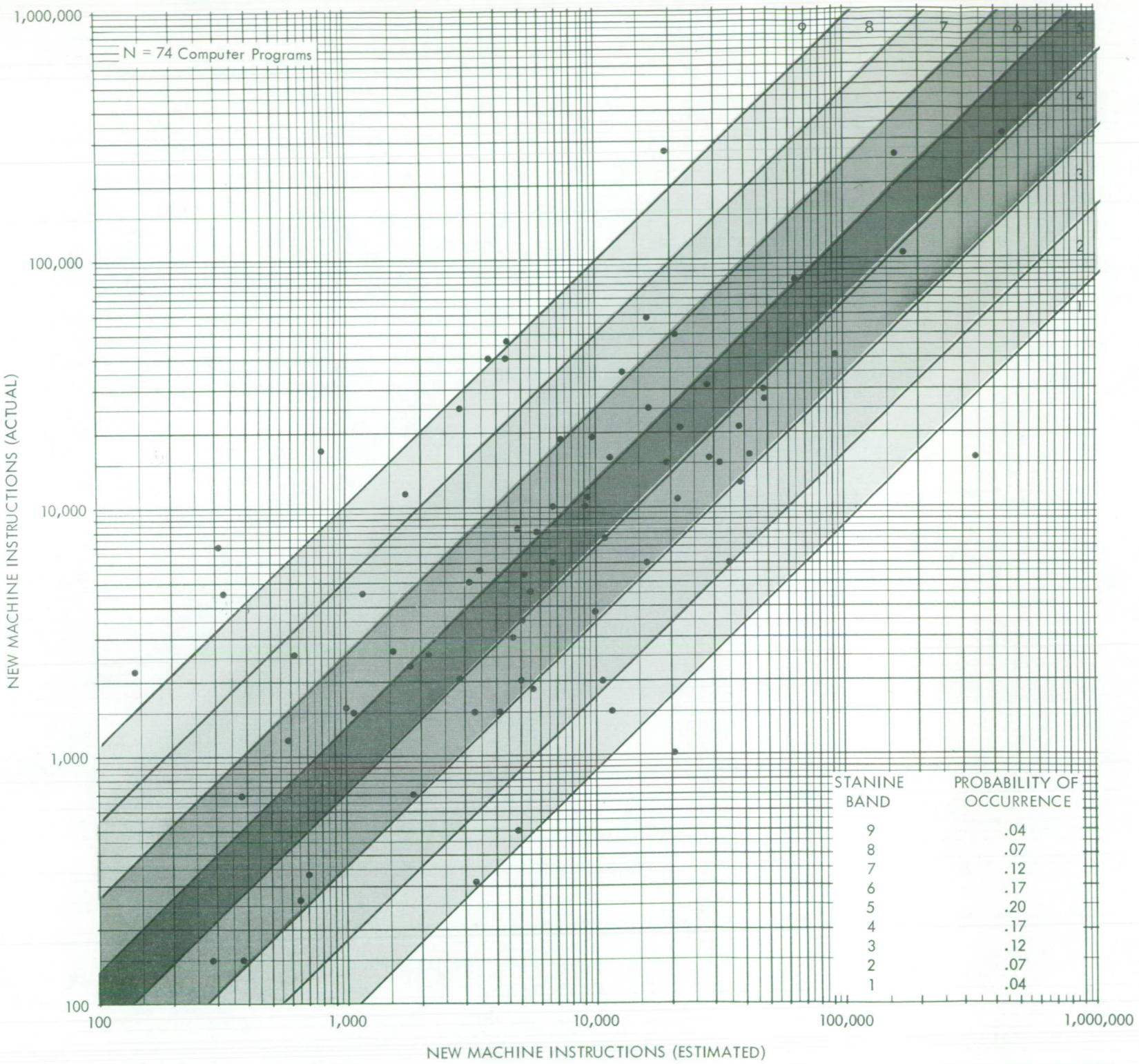


FIGURE 17

STANINE PLOT FOR NEW INSTRUCTIONS

(Page 80 Blank)

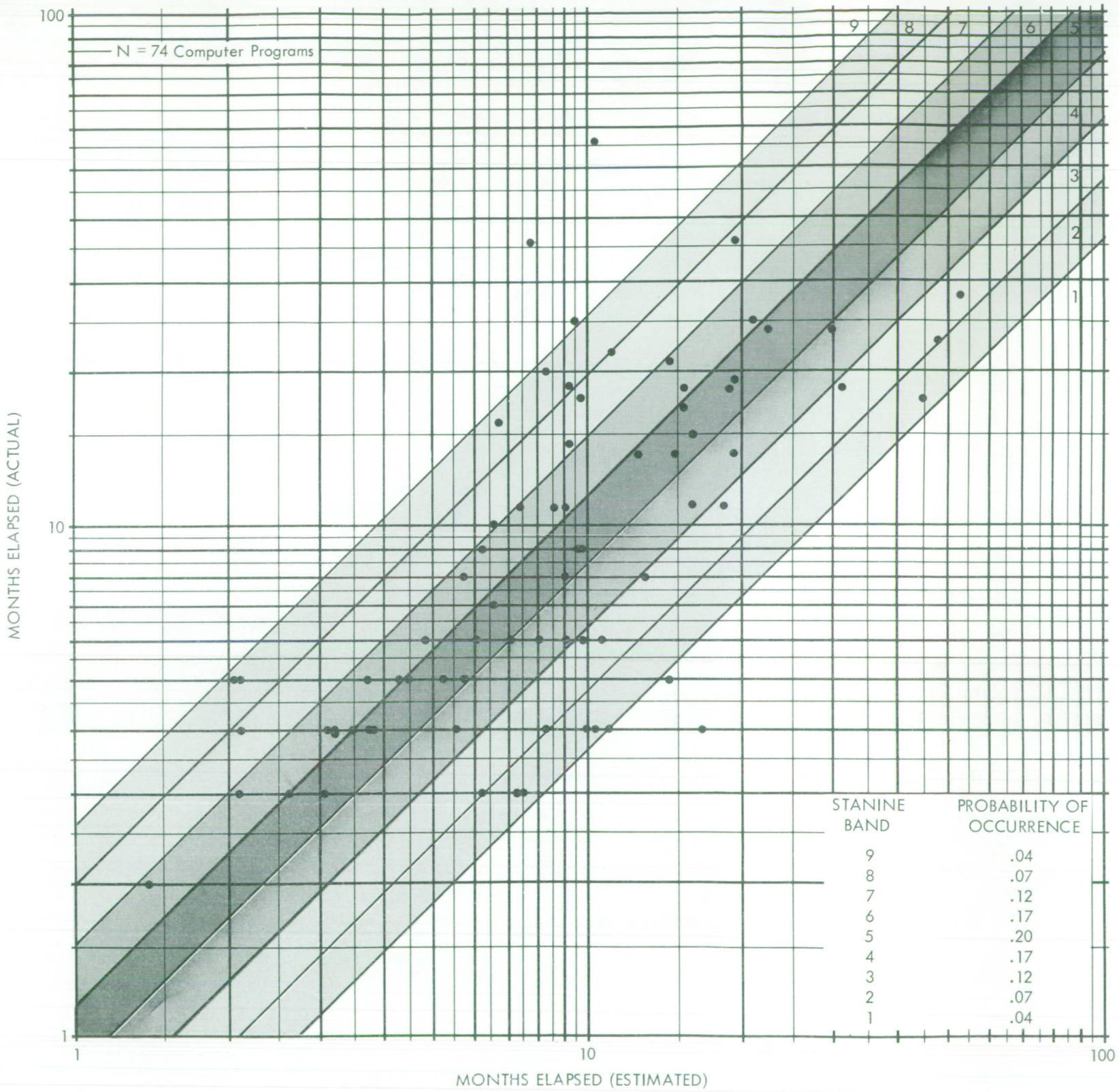


FIGURE 18

STANINE PLOT FOR MONTHS ELAPSED

(Page 82 Blank)

c. To hedge his estimation, the user can sum the probabilities of occurrence from different bands. For example, the 0.54 confidence limits for 100 man months would be 40 and 230 man months. Another approach would be to assume that errors due to underestimation are more likely than overestimation. In this case, the "5"- and "6"-band limits of 73 and 230 man months would be expected to contain twice the usual proportion of observed values (i.e., at least 54 percent), since occurrences in bands "4" or lower are not considered.[12]

The wideness of the limits is due mainly to two causes: first, the fact that all the types of programs represented in the data base were analyzed together, i.e., no subpopulations were considered separately. We intend to explore the effects of subpopulations in subsequent analyses, and hope to tighten the confidence-band limits as a result. The second reason for the rather wide bands is due to more fundamental matters involving the combination of before and after-the-fact probabilities. This question is discussed in the following Section.

The use of Figures 16, 17, and 18, dealing with computer hours, new instructions, and months elapsed, is similar.[13]

---

[12] These approaches are comparable in principle to the one- and two-tail tests common to statistical inference.

[13] We would like to point out that the straight-line confidence bands shown in Figures 14 through 18 are approximations. The bands actually diverge, as the variables involved in the estimation depart from their mean and the prediction line, i.e., the corrections, are added to the estimate above the line, and subtracted from it below the line. The corrections are unique for each value of each predictor, however, and rather laborious. A worksheet is provided in Appendix XI for readers who wish to compute the corrections for each application of the estimating equations. Two examples worked out in the Appendix illustrate the magnitude of the corrections, which are ±13 percent for the "2" and "8" bands.

SECTION XVIII

AN EVALUATION OF THE STANINE COST ESTIMATION RELATIONSHIPS

As we have observed earlier, we recommend the use of the cost estimation equations and Stanine plots as guides and checks for the manager of computer programming. In no sense are they yet sufficiently adequate or reliable techniques for the prediction of resource-costs in terms of a particular job. There are several reasons why this is so.

First, the limits of the confidence bands are still too far apart to be useful to a manager for prediction. Based on "5" band, or "average" task, Figure 15 indicates that the range of values that can be expected around an estimate of 100 man months (a computed logarithm of 2.0) lies between 73 and 130 man months. This probably represents close to the widest interval in which a manager would be interested, i.e., about plus or minus 30 percent. The addition of more bands would, of course, result in even more disparate limits.

The very wideness of the confidence intervals, which lessens the value of the plots for prediction, also increases their usefulness as a check on estimates made by other means, i.e., values that fall outside the expected bands are especially suspicious. For a relatively small, simple programming task, an estimate that falls into the high "7" band when compared to the value computed from the equations would reasonably be subject to review.

Similarly, for a programming task that is expected to be sizeable and complex, costliness estimates that fall into the low "3" or "2" bands when compared to the estimates computed from the equation would quite rightly be considered as unusual, and subjected to further scrutiny.

The aim of future research in this area is: (1) to broaden the size and representativeness of the data base from which the estimating relationships are derived; (2) to define and study subpopulations separately; (3) to improve our understanding of the programming process; (4) by all of these approaches, to narrow the range of expected variation around a computed estimate. In this context, the current collections of data from various government and industry sources are a step in the right direction.

A more substantial problem for future research is to find a more reliable basis for blending the manager's intuitive and experiential estimates with the statistical results. The Stanine plots certainly represent a practical approach that we believe is more suited to general application than the techniques that we have used previously. But the so-called "probability of occurrence" associated with each Stanine band (0.20 for the "5" band, 0.17 for the "4" and "6" bands, etc.) reflects the proportion of all computer programming jobs that likely will fall into each band. That is not what the manager is interested in. He wants to know the probability that, for a particular program, the

final resource-costs will fall within the limits prescribed by a given Stanine band, or a combination of bands. (In somewhat this sense, the Stanine probability of occurrence" represents a lower limit.)

If a manager considers a programming job to be "average," and if he is right, the probability that the resource-costs will fall within the limits denoted by the "5" band exceeds the 0.20 minimum, and could go as high as 1.00. What is needed to make the probabilities more realistic is a defensible way of combining a manager's subjective estimation of the band within which a job will fall with the theoretical probability of occurrence.

At the present state-of-the-art, we feel that, unless a manager has strong indications that a programming task is unusually simple or difficult, the "5" band values should be used. We also expect that for most programs, the chances of obtaining results within the "5" band limits are one in two, or better. But we have no data by which these assertions can be defended and by which, for instance, a manager's subjective confidence of 0.80 that a job is properly represented in the "5" band can be reliably combined with the 0.20 theoretical minimum probability of occurrence to obtain a compound probability that ultimate program costs will fall within the prescribed limits.

The long-term alleviation of this problem lies mainly in the on-line data collection that we will be undertaking. In addition to obtaining data about the task, the environment, the staff, the resources and expenditures, etc., for a sample of actual programming jobs, we intend to record the manager's a priori estimate of the Stanine band that he considers appropriate, and, by comparing it with the after-the-fact costs, derive an empirical function that represents the degree to which subjective estimates should be considered, i.e., weighted, for different types of program products.

Eventually (and this will take longer), we hope to use the on-line data to derive relationships that can be used to correct a priori estimates on the basis of actual costs to date. This approach will carry us into considerations of what are called Bayesian inferences, i.e., procedures with prior and posterior probabilities that permit the use of the latest factual information to correct earlier estimates. A good deal of data collection and analysis lies between our present capability and understanding and the realization of the potential that is inherent in the Stanine plots. Our plans into and beyond the on-line data collection and analysis, however, are deliberately aimed toward the development of these and similar techniques for management use in computer programming.

(Page 86 Blank)

SECTION XIX

THE SENSITIVITY OF PROGRAMMING RESOURCE-COSTS TO CHANGES IN THE TASK INDICES

1. Background. The equations that were developed in the previous Section have two uses: the most important, of course, is the estimation of resource-costs for programming jobs. In addition, the weights that are given each index may be used to infer their relative impact on the dependent variable. This would guide the programming manager's decisions as to planning trade-offs, e.g., to reduce total resource-costs, is it more advantageous to reduce the "difficulty" of the job or improve the "development environment," as we measure them?

Unfortunately, the equation coefficients for each index are not susceptible to direct interpretation. One reason is that some of the predictors that enter into the indices are in raw form while others are logarithms or percentages. The weight given to each index tends therefore to reflect the importance of the combination of the variables as they appear in the equation, and not as the manager considers them, i.e., in raw form.

Another obstacle to direct interpretation of the regression coefficients is that some of the indices vary through a much wider range than others. An index with a relatively low weight but a large variance may exert a greater impact on resource-costs than an index with a high weight and little variation. For instance, the Uniqueness Index, which is composed of five binary predictors, varies through a much smaller range than does the Job Difficulty Index, which has no upper limit.

2. The Limitations of this Approach. A formal sensitivity analysis(21) requires a great deal more data than are available in our present 74-point base. In this sense, the material in this Section is exploratory, and is included to illustrate the application of the sensitivity concept. We expect to develop more trenchant and reliable measures of the relative impact of the indices in subsequent analyses, as the data base is enlarged and refined.

3. The Analytical Procedures and Results. To determine the sensitivity of each of three resource-cost variables, Man Months, Computer Hours, and Months Elapsed, values for them were computed from the estimation equations, with each of the indices at their respective means. Next, for each equation (i.e., dependent variable) at a time, the indices were iteratively changed one at a time while the remaining three indices were held at their means. Four variations were applied to each index: +10 percent, +1 standard deviation, -10 percent, and -1 standard deviation. The difference between the effect on

resource-costs of the 10-percent and 1-standard-deviation changes is intended to illustrate the impact of the wide variations that are possible in certain indices.

The results of these procedures are illustrated in Figures 19, 20, and 21. The actual values are provided in tabular form in Appendix XII.[14]

4. Interpretation of the Results. Several general conclusions can be made from a consideration of Figures 19, 20, and 21:

a. The Uniqueness and Job Difficulty Indices have a very substantial impact on the estimation for Total Computer Hours. Increasing either index by 1 standard deviation from its mean essentially doubles the estimate for this resource-cost. (Reducing the Uniqueness and Job Difficulty Indices by 1 standard deviation from their mean reduces the estimate for Computer Hours by about one-half. The asymmetry is due to the nature of the data that were used in the analysis.) Increasing or decreasing the Uniqueness and Job Difficulty Indices by 1 standard deviation from their means has a lesser but still significant impact on the estimates for Total Man Months and Months Elapsed, e.g., changes in the estimate from 22 to 68 percent result.

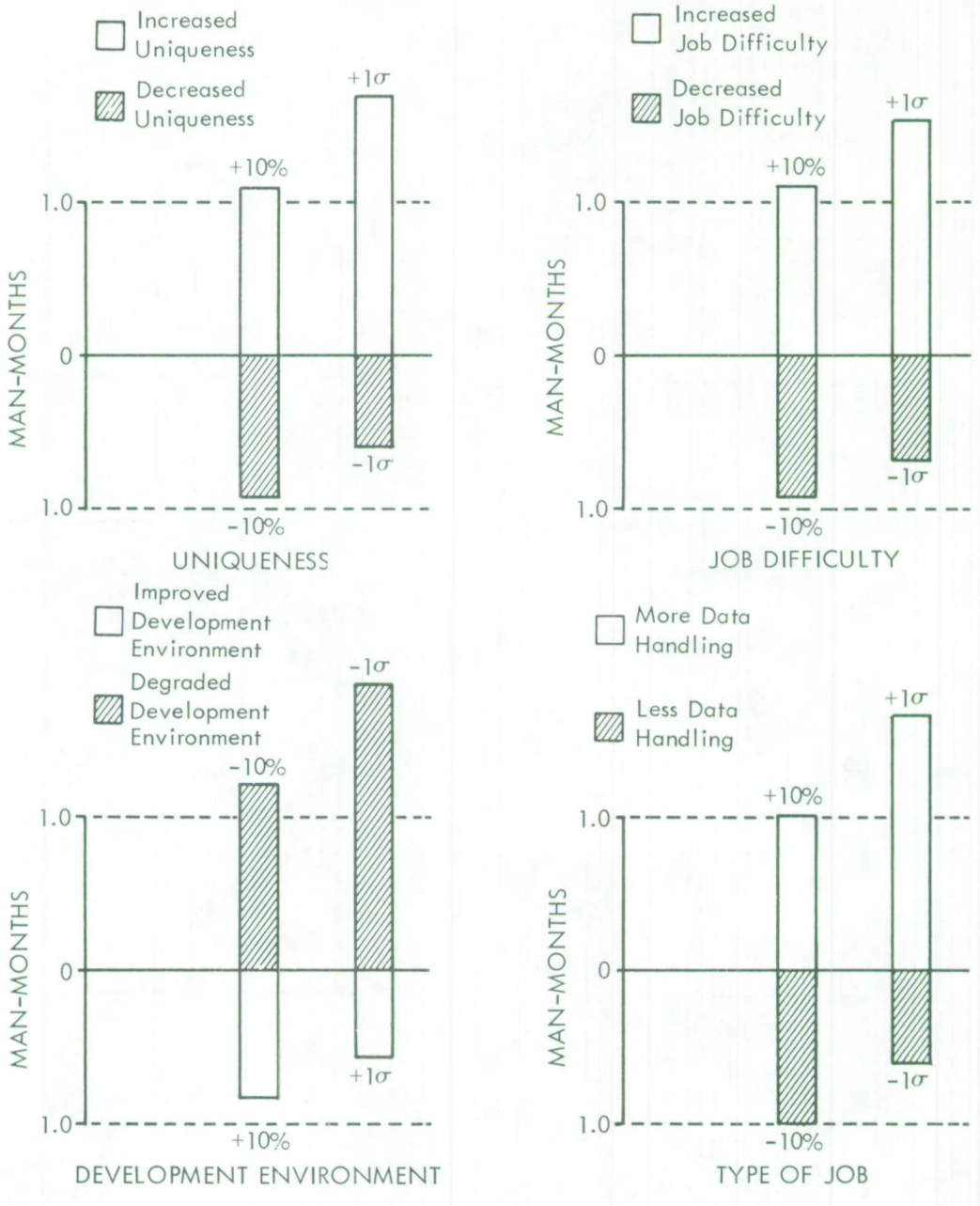
b. The Development Environment Index has a considerable effect on the estimation for Total Man Months and Total Computer Hours. Changes of 10 percent from the mean in this index result in estimate changes of from 15 to 19 percent in the dependent variables. Changes of 1 standard deviation from the mean result in estimate changes of from 43 to 83 percent in Total Man Months and Total Computer Hours.

c. Small changes in the Job Type Index have little impact on estimates for the three resource-cost variables under consideration. Due to its wide variance, however, changes in this index of 1 standard deviation from the mean cause changes in the estimates of from 13 to 66 percent in the dependent variables.

d. Months Elapsed is about one-half as sensitive to changes in the indices as Total Man Months and Total Computer Hours. This may be a reflection of the fact that managers tend to satisfy schedule constraints by making compensatory changes in Total Man Months and Total Computer Hours.

---

[14] Note that the Development Environment Index (variable 151) decreases as resource-costs increase, and vice versa. This occurs because the components of the index are negatively weighted to be consistent with intuitive expectation, i.e., so that improvements in the environment (larger negative index values) decrease resource-costs.

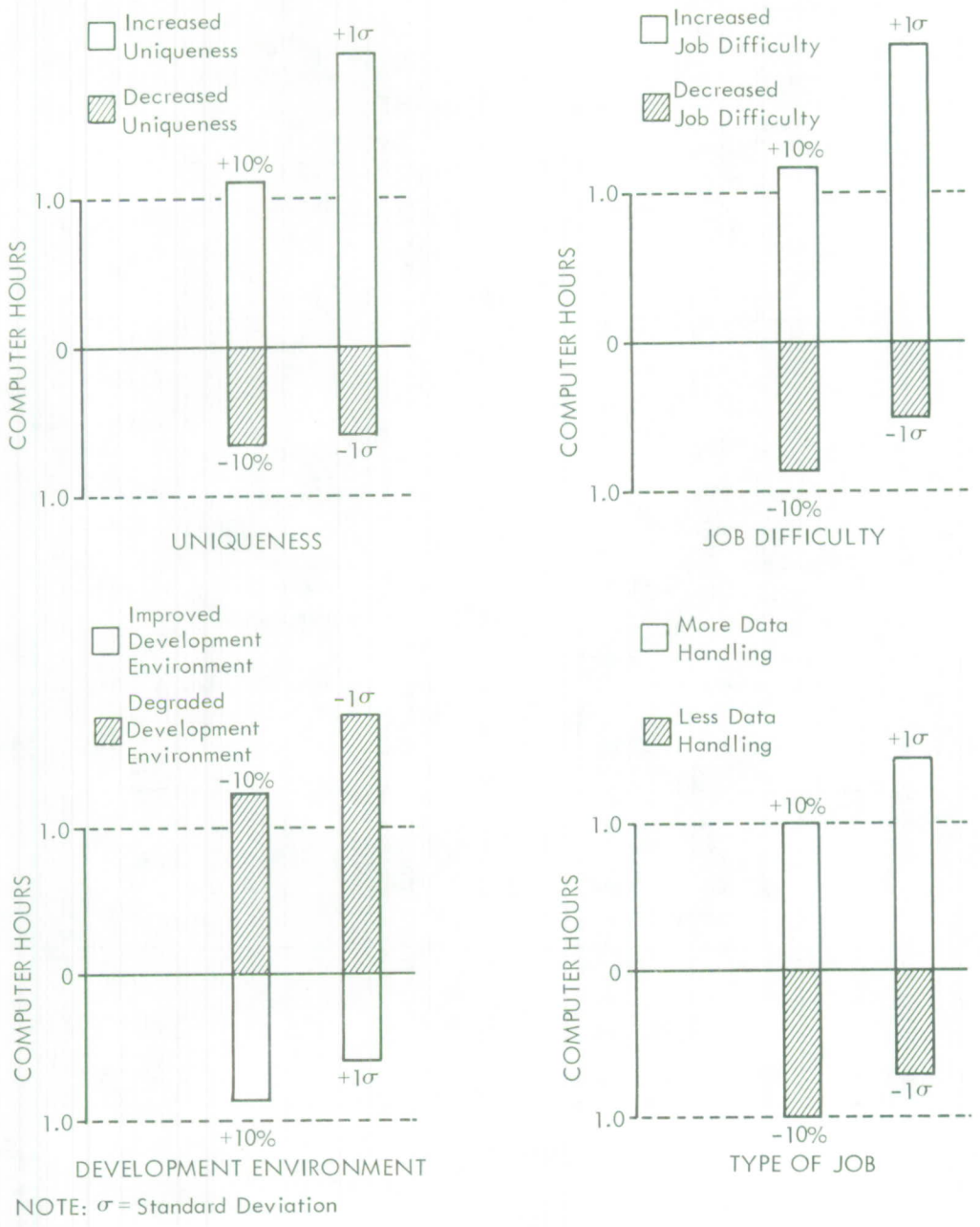


NOTE: σ = Standard Deviation

FIGURE 19

PERCENT CHANGE IN MAN MONTHS

DUE TO INDICATED CHANGE FROM THE INDEX MEAN

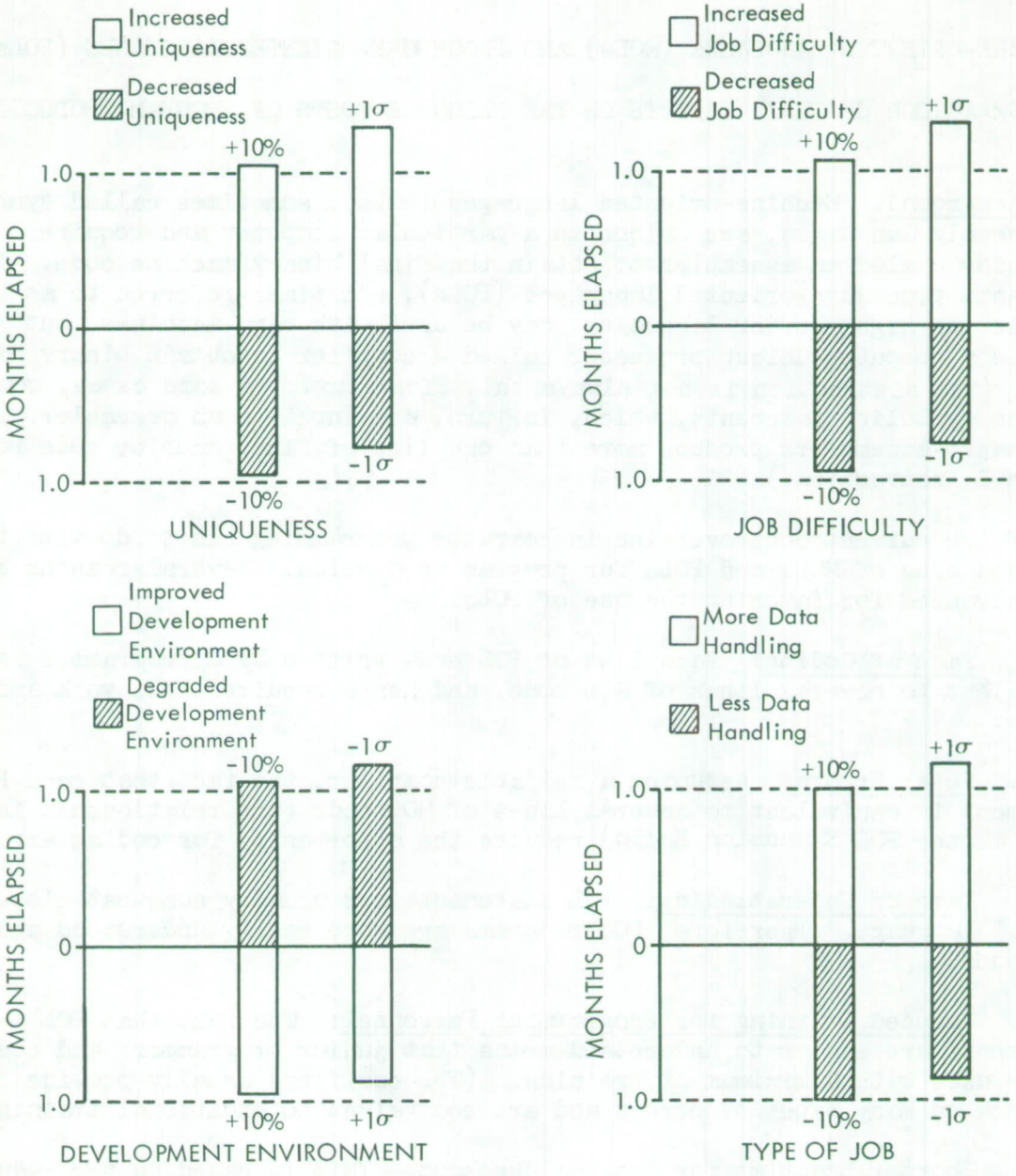


NOTE: σ = Standard Deviation

FIGURE 20

PERCENT CHANGE IN COMPUTER HOURS

DUE TO INDICATED CHANGE FROM THE INDEX MEAN



NOTE: σ = Standard Deviation

FIGURE 21

PERCENT CHANGE IN MONTHS ELAPSED

DUE TO INDICATED CHANGE FROM THE INDEX MEAN

SECTION XX

MACHINE-ORIENTED LANGUAGES (MOLs) AND PROCEDURE-ORIENTED LANGUAGES (POLs): A COMPARISON OF THEIR EFFECTS ON THE RESOURCE-COSTS OF PROGRAM PRODUCTION

1. Background. Machine-oriented languages (MOLs), sometimes called symbolic or assembly languages, are unique to a particular computer and require a processor called an assembler to obtain the final binary machine code. By contrast, procedure-oriented languages (POLs), sometimes referred to as compiler or higher-order languages, may be used with many machines, but require a computer-unique processor called a compiler to obtain binary machine code. (The distinction is not always this clear-cut. In some cases, compilers produce symbolic statements, which, in turn, are input to an assembler. Also, some macro-assemblers produce more than one line of final machine code for each MOL instruction.)

One of the current controversies in computer programming has to do with the pros and cons of MOLs and POLs for program production. Several reasons have been advanced for favoring the use of POLs:

a. Ease of Coding: Each line of POL code written by a programmer is equivalent to several lines of MOL code, and hence requires less work and review.

b. Fewer Errors: Assuming a reliable compiler, the fact that each POL statement is equivalent to several lines of MOL code (the relationship is known as the POL Expansion Ratio) reduces the opportunity for coding errors.

c. Ease of Understanding: POL statements are usually somewhat closer to natural language. Therefore, POL programs are more easily understood and discussed.

d. Reduced Training for Programming Personnel: The fact that POL statements are easier to understand means that junior programmers and coders can be used with a minimum of training. (The compilers usually provide feedback on many types of errors and are equivalent to additional training.)

e. Shorter Leadtime for Program Checkout: This is based on the reduction in the rate of errors and the relative ease in making changes.

f. Reduced Need for Documentation: The fact that representation in a POL is somewhat closer to natural language may eliminate some separately-written program design documentation.

g. Ease of Program Conversion: The machine-independent nature of POLs eases the conversion of programs from one computer to another, assuming that the necessary compiler is available.

The MOL advocates, in turn, are critical of POLs for several reasons:

a. Obtaining final machine code through compilers may consume more computer time than does the use of assemblers.

b. Final machine code produced by compilers tends to be somewhat longer (looser) than that produced by assemblers, for equivalent programming tasks.

c. For somewhat the same reasons, the final machine code produced by compilers may take longer to operate than does the code produced by assemblers, for equivalent programming tasks.

d. The compiler costs more time and money to develop than does an assembler.

These pros and cons have been debated and "debunked" by a variety of opinion-leaders in the ADP field(22). Few facts and fewer numerical data are available to help settle these controversies (and, in some cases, the few data that have been available have, at times, been misused to promote the aims of a particular advocate).

While the 60 MOL and 14 POL data points that are represented in our present data base are too few and some specific information is lacking conclusively to confirm or refute these claims and counterclaims, we have made some preliminary comparisons in certain areas.

2. Analytical Procedures. The comparisons made in this Section were between 14 programs coded in a single POL--JOVIAL--and 60 programs produced in a variety of MOLs used by the System Development Corporation.

The comparisons were made in terms of four ratios developed in the course of this analysis.[15] Briefly, the ratios indicate the following:

a. The Program Production Rate represents the average net instructions delivered per man month expended on program design, code, and test. (This is not to be confused with the average amount of net code that a programmer will deliver in a month, apart from the design and test phases.)

---

[15] For these MOL and POL comparisons only, raw values of the ratios were used. Elsewhere in the analysis, the values for the first three of these ratios were considered after being logarithmically transformed.

b. The Computer Usage Rate represents the average number of computer hours expended for each instruction coded and delivered.

c. The Documentation Rate represents the average number of pages of documentation produced for each instruction coded and delivered.

d. The POL Expansion Ratio represents the average number of words of machine code that are compiled from each POL statement.[16]

Since POLs, by their nature, require fewer statements than the resulting number of machine code words after compilation (the POL Expansion Ratio measures this characteristic), it was necessary to define a common product in terms of which resource expenditures could be compared. This was done by considering the newly coded portion of the delivered sequence of machine instructions as the common standard, produced in one case by a MOL and in the other via a POL. The definitions of the four ratios and their components are provided in Table XIII.

Since we intended to compare the means of the four ratios for programs produced via MOLs and POLs, it was important that the means were not unduly influenced by extreme values (outliers). This is especially the case when working with very small samples, such as the 14 POL points. Relying on well-established procedures(10), we marked points that were more than three standard deviations from the median, and replaced them by values at 3.0 and 2.9 standard deviations, respectively, in order of their rank. The number of data adjusted in this manner, out of 60 MOL and 14 POL points, is shown in Table XIV.

TABLE XIV

NUMBER OF MOL AND POL POINTS ADJUSTED

		Low	High
Production Rate:	MOL	0	1
	POL	0	0
Computer Usage Rate:	MOL	0	2
	POL	0	0
Documentation Rate:	MOL	0	2
	POL	0	0
POL Expansion Ratio:	POL	0	0

[16] Since no macro-assemblers were used in the production of the 60 MOL programs represented in the present data base, there is no MOL expansion ratio, i.e., each MOL instruction resulted in one word of final machine code.

TABLE XIII

DEFINITION OF VARIABLES FOR MOL VERSUS POL COMPARISONS

MOL Net Instructions Coded	=	(New MOL instructions in delivered program) + (Nonfree, nondelivered MOL utility instructions)
POL Net Equivalent Machine Instructions	=	(New MOL machine instructions in delivered programs, after compilation) + (MOL instructions in nonfree, nondelivered utility programs, after compilation) - (MOL instructions inserted directly in either of the previous portions of the programming task)
Total Man Months	=	Man months to design, code, and test the delivered program
Primary Computer Hours	=	Aggregate hours of primary computer operate time used to the end of the program test phase
Total Pages Documentation	=	Published pages (internal + external)
MOL production Rate	=	$\frac{\text{MOL Net Instructions Coded}}{\text{Total Man Months}}$
POL production Rate	=	$\frac{\text{POL Net Equivalent Machine Instructions}}{\text{Total Man Months}}$
MOL Computer Usage Rate	=	$\frac{\text{Primary Computer Hours}}{\text{MOL Net Instructions Coded}}$
POL Computer Usage Rate	=	$\frac{\text{Primary Computer Hours}}{\text{POL Net Equivalent Machine Instructions}}$
MOL Documentation Rate	=	$\frac{\text{Total Pages Documentation}}{\text{MOL Net Instructions Coded}}$
POL Documentation Rate	=	$\frac{\text{Total Pages Documentation}}{\text{POL Net Equivalent Machine Instructions}}$
POL Expansion Ratio	=	$\frac{\text{POL Instructions Delivered}}{\text{Equivalent Machine Instructions}}$

Considering the fact that the MOL and POL data were not ideally suited for comparisons of this type, i.e., the POL sample was quite small and the samples disparate (14 and 60), some skewness was apparent for both the MOL and POL data, we used the Student "t" statistic as an appropriately conservative way[17] to compute the comparisons shown in Table XV.

3. Interpretation of the Results. Since all the data upon which these comparisons were based were taken from the one organization, the System Development Corporation, and represent one POL, JOVIAL, the results of the analysis cannot necessarily be thought to reflect either POLs or programming generally.

On the other hand, the substantial difference in favor of POLs[18] suggests rather strongly that they do require significantly less manpower, computer time, and documentation than would the same jobs coded via a MOL.

Unfortunately, we do not have sufficient data to test others of the claims and counterclaims that were summarized earlier in this Section, e.g., whether POLs permit programmers with less training to be used, to what extent compiler design can mitigate the comparisons, etc. We intend to pursue these and related questions as the data becomes available.

---

[17] This is a rather abstract technical point, but, in statistical theory, each piece of data is considered as the mean of the distribution that would result if every program were produced in the same way a large number of times. Since we can say very little about the characteristics of such distributions (since no one repeats programming jobs to collect this sort of data), the Student "t" statistic will contain terms reflecting this uncertainty. These terms will inflate the variation that enters into the statistic and thereby degrade the confidence we have in our inferences. In fact, then, the significances of the differences between the ratio means as shown in Table XIV are quite conservative, i.e., the probability of the differences occurring by chance may be much less, but we do not have the data to demonstrate this case.

[18] Some experts say that the JOVIAL compiler produces programs that are 10 to 15 percent longer, in final machine code, than if they had been written in a MOL. While we did not correct for this, the magnitude of the ratio differences is large enough that the implications would not have changed significantly.

TABLE XV

MOL AND POL COMPARISONS

	STATISTIC		LIMIT OF THE LOWER CONFIDENCE BAND FOR THE POPULATION MEAN[19]	SAMPLE MEANS	LIMIT OF THE UPPER CONFIDENCE BAND FOR THE POPULATION MEAN[19]	PROBABILITY OF THE DIFFERENCE BETWEEN MOL AND POL MEANS OCCURRING BY CHANCE
Production Rate	Net Machine Instructions / Man Months	MOL	285	322	361	.05
		POL	398	555	712	
Computer Usage Rate	Hours / 1000 Net Machine Instructions	MOL	20.8	23.6	26.5	.10
		POL	9.1	12.4	15.6	
Documentation Rate	Pages / 1000 Net Machine Instructions	MOL	70.7	88.1	105.6	.20
		POL	22.9	40.2	57.4	
POL Expansion Ratio	Machine Instructions / POL Statements		3.2	3.7	4.2	-

[19] These are upper and lower confidence bands for which the probability of equally-sized-samples occurring by chance is 0.15, in each band. Discrepancies between the upper and lower bands are due to rounding.

(Page 98 Blank)

APPENDIX I

THE DATA COLLECTION QUESTIONNAIRE

The data from 74 computer programs in the final base for this analysis were collected in one of two ways. For 51 programs, the questionnaire shown in this Appendix was completed. The remaining 23 programs were carried over from the previous analysis and questionnaire (TM-1447/001/00), and required the use of a supplementary questionnaire to make the two portions of data base equivalent. The questions that constituted the supplement are marked by a black square, ■, in this Appendix.

APPENDIX I (Cont'd)

GENERAL INSTRUCTIONS

This questionnaire is a means for collecting data on completed programming efforts. These data will help us to identify and verify key factors affecting the cost of computer programs. We are seeking to increase the reliability of techniques for estimating costs of future program development.

The questionnaire is organized into seven parts so that each part may be easily delegated. The first part summarizes the major costs of the program being examined in terms of man months, computer hours, and calendar time involved. The six remaining parts are questions concerning factors that may affect the cost of computer programs and are organized as follows:

A. Operational Requirements and Design
B. Program Design and Production
C. Data Processing Equipment
D. Programming Personnel
E. Management Procedures
F. Development Environment

Generally speaking, the first two categories address the question, "What was the job to be done?" The next two ask, "What were the available resources?" and the last two examine, "What was the nature of the working environment?"

The information we are seeking is fairly detailed and will require some effort to compile. To minimize this effort, we have attempted to make the questions as clear and definitive as possible. Some of the questions may be answered readily, whereas others required that we develop definitions before these questions could be understood. We encourage answering all questions even if a best guess or estimate is all that is available. When there is some question about the accuracy of a numerical answer, give the range of values about which the midpoint is the most probable value.

In order that the programs we sample be homogeneous in at least one dimension, we have defined the bounds on each program to be examined. A program satisfying the following definition is termed a program data point.

A program data point is the smallest set of computer program instructions

(1) whose purpose is defined by someone other than the programmer,

(2) that is delivered to the user (customer) as a package, and

(3) that is loaded into the computer as a program unit or system to achieve the stated purpose or objective.

One complete questionnaire is required for each program corresponding to a data point. We will need your help in identifying data points in accordance with the above definition. By this definition, a program data point can be an operational program, a utility program, or even an experimental program. These are clearly not limited to any specific function. Similarly, the user of the program (represented by the data point) may be the buyer, but he may also be another programmer, as in the case of a utility program. The responder must keep in mind at all times the portion of the program system that he is calling the program data point when answering the questions. Additionally, the definition of a program data point includes the following limits on the scope of activities considered to be the programming process. Here, we are concerned with the activities of program design, code, test, and documentation.

Finally, we encourage you to note any qualifying remarks, or discussion concerning your answers on the reverse side of the answer sheet. Similarly, we would appreciate any comments or suggestions concerning the nature of the questions themselves.

APPENDIX I (Cont'd)

SUMMARY OF COSTS

1. Total number of man months* to program design, code, test, and document this program, including first-line supervision. Do not include the cost of any associated executive or utility program. _____

2. Number of man months to develop utility programs. _____

3. Maximum number of programmers employed on this program. _____

4. Number of months that more than 90% of the maximum number of programmers were employed. _____

5. Start date for program design. _____ Month _____ Year

   By program design, we mean the activity whose inputs are the operating system description and operational specifications and whose outputs are program design specifications and flow charts.

6. Completion date for program delivery. _____ Month _____ Year

   By program delivery, we mean the point at which the program is ready to be installed in the operational computer and begin its system test in the environment for which the program was designed.

7. Total number of computer hours used. _____ Type _____ Hours

8. a. Number of man trips for briefings, problem solving, concurrence, etc. _____

   b. What was the average round-trip distance/trip? _____

*Indicate effective man months for analysts, programmers, and support personnel.

A. OPERATIONAL REQUIREMENTS AND DESIGN*

Check Appropriately

1. Was there a requirement for innovation in the information processing system? Yes _____ No _____

   By innovation, we mean either a new data-processing application of a known programming technique and/or a new technique for a known application. By new, we mean new to the people involved.

2. Was there participation by the programming organization in the requirements analysis and/or operational design? Yes _____ No _____

   The requirements analysis is conducted to specify in detail the performance requirements of this information-processing system. These performance requirements are the input to the operational design activity, which translates the requirements into operational design specifications. These specifications indicate how the information-processing needs will be satisfied.

3. How well were the operational requirements known and documented?

   In great detail _____

   In broad outline _____

   Only vaguely _____

4. How many organizational users supply inputs and request outputs from this program? _____

5. How many automatic data-processing centers are in the system? _____

   Here we mean computer-based centers with automatic digital communications.

---

*Part A pertains to the information-processing system in which the computer system is embedded. Part B is concerned with program design considerations.

■ 6. Characterize the complexity of the overall information-processing system on a scale of (1)-(5) from simple to highly complex. _____

While almost all of the previous questions in this section relate to the complexity of the information-processing system, this question attempts to aggregate these factors into one index of complexity. Select that descriptor that most nearly applies.

Complexity Scale

(1) Simple operational functions and few users. Direct translation of manual tasks to automatic functions. Off-line output with no interfacing agencies.

(2) Operational requirements fairly well understood. Only a few new tasks to be automated. No elements of on-line or real-time operations.

(3) Operational requirements known in only a broad outline. Operational functions to be performed require some analysis to determine how best to design information-processing system. System may or may not be real time, has few interfacing agencies providing input and requiring output.

(4) Operational requirements known only vaguely. Many new functions requiring decision-making capabilities. Mixture of on-line and off-line responses to many diverse users.

(5) Many new operational functions, hardware components to be developed and interacting users. Operational requirements highly unstructured and indeterminate. System must have great capability for reacting to new situations on real time basis.

B. PROGRAM DESIGN AND PRODUCTION

■ 1. Indicate the number of computer program instructions and words. EXCLUDE the contents of the data base and tables used for storage of input and output messages and interprogram communications.

By instructions, we mean machine language instructions or orders. If the computer is a multiaddress machine, count each instruction separately. If a procedure-oriented language (POL) is used, list the number of POL statements in parenthesis alongside the resulting machine instructions.

By subprogram, we mean the first level in the logical subdivision of data-processing functions in the program being considered.

By subroutine, we mean some well defined, logical or mathematical function.

■ a. Total instructions in delivered program. _____

- New instructions written for this program system: in subroutines, logical blocks, subprograms inserted in programmed portions. _____
- Reused instructions: in subroutines, logical blocks, and subprograms obtained from previous versions of this program or libraries. _____

■ b. Total instructions written but discarded and not delivered. _____

- Number of instructions discarded due to operational changes. _____
- Number of instructions discarded due to error corrections. _____

c. Number of subprograms in this program system. _____

2. Total number of logical words (items) in the data base. _____

By data base, we mean the subset of tables that describe the environment of the problem that the program is solving and/or the files to be processed. Here, if the data base changes in size relatively often, indicate an average size.

3. Total number of classes of items in the data base. _____

By classes, we mean categories or types of items such as names of people, salaries, cities, states, or any characteristics of information for which there are many items or entries.

4. Total number of words in tables and constants not included in the data base. _____

5. Indicate the number of message types handled by the program by:

Input message types _____

Output message types _____

By message type, we are not concerned here with the equipment involved, but are interested in the nature of the input message, such as flight plan, position report or sensor data, and output message types such as displays and reports.

6. Characterize the complexity of the program design by the scale (1) to (5). Select the descriptor that most nearly applies. _____

Complexity Scale

(1) Initial program design carried through without change; subprograms are nearly independent, like closed subroutines; intercommunications between programs at a minimum; no storage and timing problem.

(2) Few changes to initial program design; programmers familiar with this type of design; interprogram communication low, no need for common data pool; less than 10% reprogrammed due to nonoperational changes.

(3) Frequent changes in program design; reassignments of personnel to assist in areas estimated too low; over 10% of effort going to solve program intercommunication problems and reprogramming; data pool considered but need marginal, might have been useful in checkout.

(4) Many changes to initial program design; personnel unfamiliar with many concepts, up to 50% of effort needed for interprogram communications, data pool needed or should have been used; over 50% reprogrammed to solve nonoperational problems, such as table redesign, errors found in checkout, etc.

(5) Initial program design completely remade; over 50% of personnel reassigned, well over half of effort goes to communications problems; storage and timing problems of the subprograms affect several other subprograms; discarded registers approach the size of the final program.

7. Characterize the nature of the data processing by indicating the percentage of the program's instructions devoted to the following:

- Clerical _____
- Mathematical _____
- Input/Output _____
- Logical-Control _____
- Self-checking, FIX _____
- Other (specify) _____

Examples

Clerical	- bookkeeping, sorting, searching, file maintenance.
Mathematical	- evaluate formula for given parameters, solve equations.
Input/Output	- format data for output equipment, translate input messages.
Logical-Control	- sequence the data-processing operations and I/O according to orders, priorities or timing requirements.
Self-Checking, FIX	- monitoring programs which detect and report errors; some try to repair the effects of generated errors.

8. Characterize the program function as percentages of one or more of the following major functions:

- information storage and retrieval _____
- data acquisition and display _____
- control or regulation _____
- decision making; choosing an optimum _____
- transformation; reformatting data _____
- generation to produce desired outputs _____
- other (specify)

9. What is the average operate time of the completed program? _____

10. How frequently does the completed program cycle or operate? (e.g., number of times/day or week) _____

11. Indicate the existence of constraints on program design, such as

- insufficient memory capacity Yes ____ No ____
- insufficient input/output capability Yes ____ No ____
- stringent timing requirements Yes ____ No ____
- other (specify) Yes ____ No ____

12. What language was used in coding the program? _____

13. The following concerns utility and support programs used in developing this program:

These are programs for generation, simulation and reduction of data; control and facility programs for modification, error correction, intervention and error detection and program for diagnostics, loading, editing and file maintenance of the system being designed.

a. Number of instructions in required support programs. _____

b. Number of distinct support programs used. _____

c. Number of support programs available without cost at the beginning of the job. _____

d. Number of instructions in the support programs available without cost. _____

14. Were there documented test plans and test designs which specified input test data and expected outputs? Yes ____ No ____

15. What was the extent of documentation?

	Internal	External
a. Number of different types of documents.	_____	_____
b. Number of pages in one set of documents.	_____	_____

By types of documents, we mean documents such as Operating System Description, Program Design Specifications, Status Reports, Error Correction Reports, Program Listings, etc.

By internal, we mean for the programming organization's use.

By external, we mean for delivery to the customer.

C. DATA PROCESSING EQUIPMENT

1. a. Give name(s) of the computer(s) used in the development of this program and size of core storage.

_____

_____

■ b. Was this your first program development effort on this computer? Yes ____ No ____

■ 2. a. On the average, what was the turnaround time experienced by the programmers?

_____

By turnaround time, we mean the total elapsed time between submission and return of a computer run.

3. How many automatic data-processing components were developed concurrently with this program?

_____

By ADP components, we mean those pieces of equipment that are recognized, addressed, or controlled by the computer program (e.g., display consoles, message composers, AD converters, etc.).

4. a. List the number and types of (CRT) display equipment driven by this program.

b. List the number and types of on-line input/output equipment (e.g., tapes, typewriters, etc.).

D. PROGRAMMING PERSONNEL

1. Fill in the following table with average* values:

	I	II	III	IV
. number of programmers by type	____	____	____	____
. years of experience with language used	____	____	____	____
. years of experience with computer used	____	____	____	____
. years of experience with this application	____	____	____	____

Type	Position	Description
I	Coder	Writes machine language instructions from flow charts. Helps prepare flow charts and test programs.
II	Programmer	Develops programs to solve well-defined problems. Prepares flow charts, writes instructions, tests programs, modifies established computer programs.
III	Senior Programmer	Conceives, develops and improves large, complex computer programs, e.g., automatic programming routines. Improves efficiency of existing programs.
IV	System Programmer	Formulates and plans new program system applications. Keeps abreast of related economic disciplines and new information processing technology. Is highly creative in designing and developing major computer program systems.

2. a. How many programmers participated in the program design? _____

b. How many programmers worked on the program for the entire duration of the project? _____

c. What was the average programmer turnover rate? _____

By turnover, we mean the number of programmers replaced per month divided by the total programmer staff.

*By average, we mean the mode, or that number that occurs most often.

E. MANAGEMENT PROCEDURES

1. Was there a documented plan or procedure for the following?

	Yes	No
a. Evaluation and implementation of system design changes.	_____	_____
b. Evaluation and implementation of program design changes.	_____	_____
c. Dissemination of error-detection and error-correction information.	_____	_____
d. Use of the computer facility, e.g., allocation of time, number of runs per day per programmer.	_____	_____
e. Contingency plan in the event the computer was overloaded or otherwise unavailable.	_____	_____
f. Communicating with other agencies.	_____	_____
g. Concurrence on design specifications.	_____	_____
h. Cost control.	_____	_____
i. Management control in the form of PERT or Gantt charts.	_____	_____
j. Document control (e.g., design file and documentation standards).	_____	_____
k. Standards for coding, flow charts, etc.	_____	_____

F. DEVELOPMENT ENVIRONMENT

1. a. State the number of agencies whose concurrence was required on operational design specifications. _____

By concurrence here, we mean the actual documented agreement and approval that the specifications are understood and satisfactory to the agency concerned. The activity of concurrence consists of preparation for briefings, delivery of briefings and meetings intended to discuss the content of the design specifications.

■ b. Estimate customer experience and knowledge concerning the development of automatic data processing systems.

Extensive ____

Limited ____

None ____

2. Was the computer operated by an agency other than the program developer? Yes ____ No ____

■ 3. Was the computer facility operated on the basis of:

Open shop _____

Closed shop _____

Time-sharing _____

4. a. Was the program developed at a site other than the operational location? Yes ____ No ____

b. Is the computer at the operational site different than the computer used for program development? Yes ____ No ____

5. Did the program development take place at more than one location during the effort? Yes ____ No ____

(Page 112 Blank)

APPENDIX II

INDEX AND CODING OF VARIABLES

In this Appendix, each variable used in the analysis is identified in at least two of three ways:

- by the part and question number, if it corresponds to a specific item in the questionnaire (Appendix I).
- by a number that is associated with each variable throughout the analysis (whether or not the variable is taken directly from the questionnaire or formed in some way, e.g., a ratio).
- by a short description of the variable (and its formation, where applicable).

The coding of each variable is also indicated.

The question numbers in the first column correspond to the seven parts of the questionnaire.

S - Summary of Major Costs

A - Operational Requirement and Design

B - Program Design and Production

C - Data Processing Equipment

D - Programming Personnel

E - Management Procedures

F - Development Environment

APPENDIX II (Cont'd)

QUESTION NUMBER			VARIABLE NUMBER	CODE
S-	1	Total man months	1	Log_{10}
	2	Utility man months	2	Log_{10}
	3	Maximum number of programmers	3	Log_{10}
	4	90% Man months	4	Log_{10}
	5	Start date	5	
	6	Completion date	6	
	7	Computer	7	
		Secondary Computer	8	
		Total computer hours	9	Log_{10}
	8a	Number of trips	10	Log_{10}
	b	Distance/trip	11	Log_{10}
A-	1	Innovation in system	12	Yes = 1 No = 0
	2	Participation by programming organization in requirement and/or operation design	13	Yes = 1 No = 0
	3	How well were operational requirements known and documented	14	3 - great detail 2 - broad outline 1 - vaguely

APPENDIX II (Cont'd)

QUESTION NUMBER			VARIABLE NUMBER	CODE
A-	4	How many users supply input to and request output from program	15	Log_{10}
	5	How many ADP centers in system	16	Log_{10}
	6	Characterize complexity of overall system	17	(1) - (5) Simple to Complex
B-	1a	Total number instructions in delivered program (machine)	18	Log_{10}
		Total number instructions in delivered program (POL)	19	Log_{10}
		Number instructions written for this program (machine)	20	Log_{10}
		Number instructions written for this program (POL)	21	Log_{10}
		Number reused instructions written for this program (machine)	22	Log_{10}
		Number reused instructions written for this program (POL)	23	Log_{10}
	1b	Total discards due to operational changes	24	Log_{10}
		Total number instructions written but discarded (machine)	25	Log_{10}
		Total number instructions written but discarded (POL)	26	Log_{10}
		Instructions discarded due to operational changes (POL)	27	Log_{10}
		Instructions discarded due to error changes (machine)	28	Log_{10}
		Instructions discarded due to error changes (POL)	29	Log_{10}

APPENDIX II (Cont'd)

QUESTION NUMBER			VARIABLE NUMBER	CODE
	1c	Number subprograms	30	Log_{10}
B-	2	Number words in data base	31	Log_{10}
	3	Number classes of items in data base	32	Log_{10}
	4	Number words in tables and constants not in data base	33	Log_{10}
	5	Number message types handled by program		
		input message	34	Log_{10}
		output message	35	Log_{10}
	6	Characterize the complexity of program design	36	(1) - (5) Simple to Complex
	7	% Clerical instructions	37	%
		% Math instructions	38	%
		% I/O instructions	39	%
		% Logical control instructions	40	%
		% Self checking fix instructions	41	%
	8	% Information retrieval	42	%
		% Data acquisition and display	43	%
		% Control or regulation	44	%

APPENDIX II (Cont'd)

QUESTION NUMBER		VARIABLE NUMBER	CODE
8	% Decision making, choosing an optimum	45	%
	% Transformation-reformatting data	46	%
	% Generation to produce desired output	47	%
9	What is average operate time of the completed program	48	Minutes
10	How frequently does the program cycle operate	49	Number times/day or week
11	Insufficient memory capacity	50	Yes = 1 No = 0
	Insufficient I/O capability	51	Yes = 1 No = 0
	Stringent timing requirement	52	Yes = 1 No = 0
12	What language was used in coding	53	
13a	Total support instructions	54	Log_{10}
13b	Number of support programs used	55	Log_{10}
13c	Number of support programs available without cost	56	Log_{10}
13d	Number of instructions in support programs available without cost	57	Log_{10}
14	Was there documentation for test plans and designs specifying input test data and expected output	58	Yes = 1 No = 0
15a	Number internal documents	59	Log_{10}
	Number external documents	60	Log_{10}

APPENDIX II (Cont'd)

QUESTION NUMBER			VARIABLE NUMBER	CODE
	15b	Number of pages internal documentation	61	Log_{10}
		Number of pages external documentation	62	Log_{10}
C-	1a	Computers used in program development and core sizes	63	--
	1b	Was this first programming effort on computer	64	Yes = 1 No = 0
	2	Average turnaround time experienced by programmers	65	Hours Log_{10}
	3	Number ADP components developed concurrently with program	66	Log_{10}
	4a	List number and types of display equipment driven by program	67	--
	4b	List number and types of on-line I/O equipment	68	--
D-	1	Number type I programmers	69	Log_{10}
		Number type II programmers	70	Log_{10}
		Number type III programmers	71	Log_{10}
		Number type IV programmers	72	Log_{10}
		Number years experience with language used--type I	73	Log_{10}
		Number years experience with language used--type II	74	Log_{10}
		Number years experience with language used--type III	75	Log_{10}
		Number years experience with language used--type IV	76	Log_{10}

QUESTION NUMBER			VARIABLE NUMBER	CODE
		Number years experience with development computer--type I	77	Log_{10}
		Number years experience with development computer--type II	78	Log_{10}
		Number years experience with development computer--type III	79	Log_{10}
		Number years experience with development computer--type IV	80	Log_{10}
		Number years experience with application--type I	81	Log_{10}
		Number years experience with application--type II	82	Log_{10}
		Number years experience with application--type III	83	Log_{10}
		Number years experience with application--type IV	84	Log_{10}
	2a	Number of programmers participating in design	85	Log_{10}
	2b	Number of programmers for entire duration of project	86	Log_{10}
	2c	Average turnover rate	87	Log_{10}
E-	1	Documented plan and procedure for:		
	1a	Evaluation and implementation of system design changes	88	Yes = 1 No = 0
	1b	Evaluation and implementation of program design changes	89	Yes = 1 No = 0
	1c	Dissemination of error detection and correction information	90	Yes = 1 No = 0
	1d	Use of computer facility	91	Yes = 1 No = 0

APPENDIX II (Cont'd)

QUESTION NUMBER			VARIABLE NUMBER	CODE
	1e	Contingency for computer unavailability	92	Yes = 1 No = 0
	1f	Communicating with other agencies	93	Yes = 1 No = 0
	1g	Concurrence on design specifications	94	Yes = 1 No = 0
	1h	Cost control	95	Yes = 1 No = 0
	1i	Management control, Gantt or PERT charts	96	Yes = 1 No = 0
	1j	Document control	97	Yes = 1 No = 0
	1k	Standards for coding, flow charts	98	Yes = 1 No = 0
F-	1a	Number of agencies whose concurrence was required on operational design	99	Log_{10}
	1b	Estimate customer experience and knowledge of development of ADP systems	100	3 - Extensive 2 - Limited 1 - Vague
	2	Was computer operated by agency other than program developer	101	Yes = 1 No = 0
	3	Was computer facility operated on basis of		
		open/closed shop	102	3 = Closed 2 = Both 1 = Open
		time sharing	103	Yes = 1 No = 0

QUESTION NUMBER		VARIABLE NUMBER	CODE
4a	Was program developed at a site other than operational location	104	Yes = 1 No = 0
4b	Is operational computer different than development computer	105	Yes = 1 No = 0
5	Did program development take place at more than one location	106	Yes = 1 No = 0
	Net machine instructions coded = New instructions + (utility instructions - free utility instructions)	107	Log_{10}
	Net POL instructions coded = New POL instructions + POL equivalent machine instructions coded	108	Log_{10}
	% Error rate (machine) = $\frac{\text{Number error discards (machine)}}{\text{(New instructions + total discards) machine}}$	109	%
	% Error rate (POL) = $\frac{\text{Number error discards}}{\text{(New instruction + total discards) POL}}$	110	%
	Production rate (machine) = $\frac{\text{Net machine instructions coded}}{\text{Total man months}}$	111	Log_{10}
	Production rate (POL) = $\frac{\text{Net POL instructions coded}}{\text{Total man months}}$	112	Log_{10}
	Utility production rate (machine) = $\frac{\text{Utility instructions (machine) - free utility instructions (machine)}}{\text{Utility man months}}$	113	Log_{10}
	Utility production rate (POL) = $\frac{\text{(POL power index(utility instructions(machine) - free utility instructions (machine)}}{\text{Utility man months}}$	114	Log_{10}

APPENDIX II (Cont'd)

QUESTION NUMBER		VARIABLE NUMBER	CODE
	% Operational discards = $\frac{\text{Operational discards}}{\text{Net instructions coded (machine)}}$	115	%
	Miles traveled = Number of trips x average miles per trip	116	Log_{10}
	Travel rate = Miles traveled per net instructions coded (machine)	117	Log_{10}
	Pages documentation = Pages internal + pages external	118	Log_{10}
	Documentation rate (machine) = $\frac{\text{Pages documented}}{\text{Net instructions coded (machine)}}$	119	Log_{10}
	Documentation rate (POL) = $\frac{\text{Pages documented}}{\text{Net instructions coded (POL)}}$	120	Log_{10}
	Months elapsed = Completion date - start date	121	Log_{10}
	Staffing stability index = $\frac{\text{Months elapsed}}{\text{Number months 90\% of programming staff employed}}$	122	Log_{10}
	Computer usage rate (machine) = $\frac{\text{Number computer hours}}{\text{Net instructions coded (machine)}}$	123	Log_{10}
	Computer usage rate (POL) = $\frac{\text{Number computer hours}}{\text{Net instructions coded (POL)}}$	124	Log_{10}
	Throughput index = $\frac{\text{Total delivered instructions}}{\text{Primary computer core size}}$	125	Log_{10}
	Data retrieval index = $\frac{\text{Total delivered instructions (machine)}}{\text{Number data base words}}$	126	Log_{10}
	Data turnover index = $\frac{\text{Number data base words}}{\text{Input + output message types}}$	127	Log_{10}
	% Senior programmers = $\frac{\text{Number type III + number type IV programmers}}{\text{Number type I + number type II + number type III + number type IV programmers}}$	128	%

APPENDIX II (Cont'd)

QUESTION NUMBER		VARIABLE NUMBER	CODE
	% Junior programmers = $\frac{\text{Number type I + number type II programmers}}{\text{Number type I + number type II + number type III + number type IV programmers}}$	129	%
	% New instructions (machine) = $\frac{\text{New instructions (machine)}}{\text{Total delivered instructions (machine)}}$	130	%
	% New instructions (POL) = $\frac{\text{New instructions (POL)}}{\text{Total delivered instructions (POL)}}$	131	%
	% Reused instructions (machine) = $\frac{\text{Number reused instructions (machine)}}{\text{Total delivered instructions (machine)}}$	132	%
	% Reused instructions (POL) = $\frac{\text{Number reused instructions (POL)}}{\text{Total delivered instructions (POL)}}$	133	%
	Management index = $\frac{\text{Sum of documentation and procedures 1's}}{\text{Total number of 1's}}$	134	--
	Direct machine instructions coded =	135	--
	POL equivalent machine instructions coded = Delivered machine instructions + utility machine instructions coded - direct machine instructions	136	
	POL Expansion Ratio = $\frac{\text{POL instructions delivered}}{\text{Equivalent machine instructions}}$	137	
	Senior man months = Total man months x % senior programmers	138	Log_{10}
	Junior man months = Total man months x % junior programmers	139	Log_{10}
	Average programmer experience = $\frac{\text{Total experience of types I + II + III + IV programmers}}{\text{Maximum number of programmers}}$	140	Log_{10}
	Average senior programmer experience = $\frac{\text{Total experience of types III + IV programmers}}{\text{Number of programmers x \% senior programmers}}$	141	Log_{10}
	Average junior programmer experience = $\frac{\text{Total experience of types I + II programmers}}{\text{Number of programmers x \% junior programmers}}$	142	Log_{10}

APPENDIX II (Cont'd)

QUESTION NUMBER		VARIABLES NUMBER	CODE
	% Programmers participating in design = $\frac{\text{Number programmers participating in design}}{\text{Maximum number of programmers}}$	143	%
	% Programmers for duration = $\frac{\text{Number programmers for duration}}{\text{Maximum number of programmers}}$	144	%
	% Utility instructions (machine) = $\frac{\text{Number utility instructions (machine)}}{\text{Total delivered instructions}}$	145	%
	Staff application experience = $\frac{\text{Total years of 4 types of programmer experience with application}}{\text{Total number of programmers - (types I-IV)}}$	146	Log_{10}
	Staff development computer experience = $\frac{\text{Total years of 4 types of programmer experience with development computer}}{\text{Total number of programmers - (types I-IV)}}$	147	Log_{10}
	Staff development language experience = $\frac{\text{Total years of 4 types of programmer experience with development language}}{\text{Total number of programmers - (types I-IV)}}$	148	Log_{10}
	Uniqueness index = 1.2 x Innovation 1.0 x Stringent timing 1.7 x First programming effort on computer 1.3 x Program developed at more than one location	149	
	Job difficulty index = 5.0 x Number of subprograms 1.0 x Number of classes in data base	150	
	Development environment index = -1.0 x Estimate customer experience -3.4 x % Programmers participating in design	151	
	Type of job index = -1.0 x % Clerical -1.3 x % Transform-reformat 4.4 x % Generation	152	

QUESTION NUMBER		"b" Weight	VARIABLE NUMBER	CODE
	Staffing index = 1.0 x % senior programmers		153	
	Costliness factor = Log_{10} number of trips	.28598	154	--
	Log_{10} pages documented	.15312		
	Log_{10} total man months	.38738		
	Log_{10} total computer hours	.20578		
	Log_{10} number new instructions	.29093		

APPENDIX III

A PARTIAL DATA BASE MATRIX

The data contained in this Appendix are a partial tabulation of responses to the data collection questionnaire and represent the variables that were considered in the formal analysis. The data presented may not be in the form that they appear in the questionnaire for the following reasons:

a. Transformation to ratios, e.g.:

$$\text{Documentation Rate} = \frac{\text{Pages Documented}}{\text{Net Instructions Coded}}$$

b. Transformation to percentages, e.g.:

$$\%\ \text{New Instructions} = \frac{\text{Number of New Instructions}}{\text{Total Delivered Instructions}}$$

c. Transformation to logarithms, e.g.:

$$\text{Log Man Months} = \text{Log}_{10}\ (\text{Total Man Months})$$

d. Formation of variable from two or more variables, e.g.:

Months Elapsed = Completion Date - Start Date

The following conventions were carried through from the computer output to this Appendix:

a. The first 60 sample points represent programs coded in machine-oriented language; the last 14 represent those coded in either procedure-oriented language, or both machine- and procedure-oriented languages.

b. The logarithm of zero is represented by the number zero; the logarithm of one is represented by a negative zero, e.g., (-0).

c. A negative entry for a logarithm represents the result of subtracting the ten's complement of the corresponding number from ten, e.g.

i.e., $\text{Log}_{10}\ 0.1400 = 9.1461 - 10.0000 = -\ .8539$

For a complete identification and definition of variables, the reader is referred to Appendix II of this Report.

APPENDIX III (Cont'd)

Sample Number	$\log_{10}$ Total Man Months (1)	$\log_{10}$ Total Computer Hours (9)	$\log_{10}$ Number of Trips (10)	Innovation in System (12)	Participation by Programming Organization (13)	How Well Were Plans Known and Documented (14)	$\log_{10}$ Number of Users Supplying Input and Requesting Output (15)	$\log_{10}$ Number of ADP Centers in System (16)	System Complexity (17)
1	-.00	1.26	0	0	0	3	-0	0.48	3
2	1.48	0.30	-0	0	1	3	0.48	0.30	2
3	0.30	0.70	0	0	0	2	-0	1.20	3
4	0.48	1.60	0	1	1	1	0.30	1.20	3
5	0.48	1.08	0	0	1	2	-0	-0	1
6	0.48	1.18	0	1	1	3	0.48	1.20	3
7	0.78	1.30	-0	0	1	3	-0	0.70	3
8	0.78	1.30	-0	0	1	3	-0	0.70	3
9	0.78	1.30	-0	0	1	3	-0	0.70	3
10	0.78	1.30	0	0	1	3	-0	-0	2
11	0.78	0.30	0	0	0	3	0.48	-0	3
12	0.78	1.58	0	0	1	3	-0	1.20	3
13	0.78	1.85	0	1	1	2	0.48	1.20	3
14	0.78	1.40	0	0	1	2	1.18	1.18	3
15	0.78	1.51	0	0	0	3	-0	0.48	3
16	0.85	1.30	-0	1	1	3	-0	0.70	3
17	1.48	1.43	-0	0	1	3	0.48	0.30	2
18	1.48	1.23	-0	0	1	3	0.48	0.30	2
19	0.90	1.30	-0	1	1	3	-0	0.70	3
20	0.90	2.20	-0	1	1	2	-0	-0	4
21	1.00	2.18	1.20	0	1	3	-0	0.70	3
22	1.08	2.30	0	1	1	2	(1.40)	(0)	3
23	1.48	1.43	-0	0	1	3	0.48	0.30	2
24	1.15	1.67	0	1	1	2	-0	1.20	3

APPENDIX III (Cont'd)

Sample Number	$\log_{10}$ Total Man Months (1)	$\log_{10}$ Total Computer Hours (9)	$\log_{10}$ Number of Trips (10)	Innovation in System (12)	Participation by Programming Organization (13)	How Well Were Plans Known and Documented (14)	$\log_{10}$ Number of Users Supplying Input and Requesting Output (15)	$\log_{10}$ Number of ADP Centers in System (16)	System Complexity (17)
25	1.20	2.08	0.70	1	1	2	0.60	0.30	3
26	1.20	1.88	0.90	0	1	3	-0	0.70	3
27	1.28	1.78	0.90	1	1	2	-0	-0	4
28	1.28	1.49	0.60	0	1	3	-0	-0	3
29	1.30	2.70	1.74	1	1	2	0.30	-0	3
30	1.38	2.18	1.20	0	1	3	-0	0.70	3
31	1.40	2.18	1.20	0	1	3	-0	0.70	5
32	1.48	2.40	0.70	1	1	3	0.78	1.23	5
33	1.65	2.13	0.48	1	1	2	2.00	-0	5
34	1.72	2.41	0.60	1	0	3	0.70	1.53	2
35	1.77	2.74	0	1	1	2	0.70	-0	5
36	1.81	2.80	0	0	1	3	-0	-0	3
37	1.89	2.78	0	1	0	1	0	-0	3
38	1.98	2.48	0.60	1	1	1	-0	0.30	4
39	2.04	2.78	1.30	0	1	1	0.78	-0	4
40	2.08	2.65	0.70	0	1	3	0.70	1.53	3
41	2.11	2.94	1.20	0	1	2	-0	0.48	3
42	2.15	2.11	0	1	1	2	0.30	0.30	3
43	2.26	2.29	1.62	0	1	2	0.30	0.30	3
44	2.26	2.76	1.40	1	1	3	0.60	-0	4
45	2.28	3.15	1.48	0	1	3	0.30	1.30	3
46	2.30	2.51	0	0	1	3	0.30	0.30	3
47	2.32	3.00	1.48	1	1	2	-0	0	3
48	2.37	2.93	1.00	0	1	1	0.30	0.30	4
49	2.40	3.11	1.18	1	1	3	0.60	-0	4

APPENDIX III (Cont'd)

Sample Number	Log_{10} Total Man Months (1)	Log_{10} Total Computer Hours (9)	Log_{10} Number of Trips (10)	Innovation in System (12)	Participation by Programming Organization (13)	How Well Were Plans Known and Documented (14)	Log_{10} Number of Users Supplying Input and Requesting Output (15)	Log_{10} Number of ADP Centers in System (16)	System Complexity (17)
50	2.41	3.21	1.32	0	0	2	-0	1.53	3
51	2.48	3.04	1.30	1	1	3	0.60	-0	4
52	2.73	3.00	2.95	1	1	2	(2.85)	0.60	5
53	2.78	3.35	1.15	1	1	2	-0	-0	4
54	3.16	3.93	1.95	1	1	2	0.60	0.30	5
55	3.16	3.96	2.13	1	1	2	0.60	0.30	5
56	3.21	3.85	0	1	1	2	0.30	0.48	4
57	1.72	2.44	0.60	1	1	3	0.70	1.53	2
58	2.08	2.65	0.70	1	1	3	0.70	1.53	3
59	1.43	2.35	0	1	1	2	0	1.15	3
60	1.70	2.00	0	0	1	1	1.00	0.30	4
61	0.48	1.48	0	0	1	2	0.48	1.20	2
62	0.48	1.18	0	0	1	1	0.48	1.20	3
63	0.60	0.85	0	1	1	1	2.3	-0	4
64	0.60	1.88	0	1	1	1	0.30	-0	3
65	0.60	1.40	0	1	1	1	0.30	-0	3
66	0.70	1.08	-0	1	1	1	0.78	-0	4
67	1.08	1.62	0.70	1	1	2	-0	-0	3
68	1.40	1.85	0.48	1	1	2	0.48	1.20	3
69	1.66	2.44	2.44	1	1	3	0.60	-0	3
70	1.82	3.32	0.48	1	1	2	0.78	-0	5
71	2.27	2.63	0	1	1	1	0.30	-0	3
72	2.79	3.29	3.26	1	1	3	0.60	-0	3
73	1.99	2.40	2.62	1	1	3	-0	-0	3
74	1.11	1.96	0.30	1	1	3	-0	-0	3

APPENDIX III (Cont'd)

Sample Number	$\log_{10}$ New Machine Instructions Written (20)	$\log_{10}$ Number of Reused Instructions (22)	$\log_{10}$ Number of Subprograms (30)	$\log_{10}$ Number of Words in Data Base (31)	$\log_{10}$ Number of Classes of Items in Data Base (32)	$\log_{10}$ Number of Words in Tables and Constants not in Data Base (33)	$\log_{10}$ Number of Input Message Types (34)	$\log_{10}$ Number of Output Message Types (35)
1	2.42	3.01	0.78	2.88	0.70	0	-0	-0
2	2.18	0	-0	(1.0)	-0	0	0	0
3	2.18	3.06	0.48	2.98	0.60	1.70	0.30	0.48
4	2.49	2.76	0.48	4.15	0.60	1.78	-0	0.60
5	2.53	1.81	0.48	4.72	1.11	1.48	-0	-0
6	3.30	2.48	0.30	3.97	1.70	3.86	0.60	0.60
7	3.18	3.31	0.78	2.30	0.78	2.70	1.30	1.30
8	3.61	3.11	0.95	2.30	0.78	2.70	1.30	1.30
9	3.18	3.38	1.08	2.30	0.78	2.70	1.30	1.30
10	3.31	2.64	0.70	5.70	1.88	0	-0	0.30
11	3.34	2.48	-0	1.85	0.70	3.28	0.90	0.78
12	3.08	3.55	0.95	3.21	0.70	2.60	0.30	0.48
13	2.70	2.00	0.30	4.20	1.36	2.00	0	1.30
14	3.02	0	1.88	3.70	0.70	0	0	1.00
15	2.84	3.51	0.60	2.30	0.30	1.65	-0	0.30
16	3.18	3.35	1.20	2.30	0.78	2.70	1.30	1.30
17	3.42	2.40	0.70	4.08	1.48	3.14	0	0.90
18	3.43	2.18	-0	4.02	1.30	3.03	0	0.48
19	3.59	0	1.04	3.60	0.78	2.70	1.30	1.30
20	3.30	0	0.95	3.70	0.85	3.60	0.90	0.60
21	3.54	3.65	1.28	3.60	0.78	3.00	2.00	2.00
22	3.76	0	1.15	(2.70)	(1.00)	0	(1.70)	-0
23	3.85	3.00	-0	4.11	1.70	3.00	1.08	1.00
24	3.65	3.89	0.60	4.01	0.48	0	0	0

APPENDIX III (Cont'd)

Sample Number	$\log_{10}$ New Machine Instructions Written (20)	$\log_{10}$ Number of Reused Instructions (22)	$\log_{10}$ Number of Subprograms (30)	$\log_{10}$ Number of Words in Data Base (31)	$\log_{10}$ Number of Classes of Items in Data Base (32)	$\log_{10}$ Number of Words in Tables and Constants not in Data Base (33)	$\log_{10}$ Number of Input Message Types (34)	$\log_{10}$ Number of Output Message Types (35)
25	3.90	3.48	1.28	3.90	0.48	0	0.30	-0
26	3.70	3.13	1.66	3.00	0.78	3.00	2.00	2.00
27	3.78	3.30	0.48	3.30	3.00	2.70	0.30	0.60
28	4.09	4.09	1.73	4.48	(1.40)	0	0	0
29	4.60	4.78	0.60	(4.82)	(0.70)	(3.00)	0	0
30	3.78	3.31	1.11	3.00	0.78	3.00	2.00	2.00
31	3.73	4.03	1.72	3.30	0.78	3.00	2.00	2.00
32	3.77	2.95	1.28	1.00	1.00	2.94	0.78	0.48
33	4.04	3.60	1.04	2.18	2.30	3.00	2.18	2.18
34	4.00	0	1.15	(3.99)	0.78	(2.18)	0.78	0.95
35	4.54	0	1.20	6.00	2.41	4.18	0.90	0.30
36	4.18	4.20	0.90	4.00	1.30	0	0.60	0.90
37	4.40	4.14	-0	(2.00)	(-0)	0	0.90	0.78
38	4.22	0	0.70	(2.56)	0.60	(2.00)	(1.32)	1.15
39	4.33	2.88	1.23	4.60	1.47	3.30	0.30	0.48
40	4.29	3.78	1.28	(4.08)	0.77	2.40	0.48	0
41	4.00	4.26	1.54	4.53	1.08	3.15	1.20	1.78
42	4.43	0	1.15	4.60	2.12	2.60	0.70	1.26
43	4.77	0	-0	5.00	(1.00)	0	0	0
44	4.32	4.15	1.56	(4.30)	0.85	(2.63)	0.48	0.48
45	4.18	4.18	1.85	4.70	1.40	(3.30)	1.49	1.49
46	4.50	4.00	1.20	6.04	2.01	3.48	0.95	1.34
47	4.61	5.11	1.99	4.34	1.38	3.00	1.40	1.76
48	4.66	0	1.32	5.98	2.02	4.40	1.00	1.70
49	4.48	3.95	1.56	4.32	0.85	3.66	0.48	0.79

APPENDIX III (Cont'd)

Sample Number	Log_{10} New Machine Instructions Written (20)	Log_{10} Number of Reused Instructions (22)	Log_{10} Number of Subprograms (30)	Log_{10} Number of Words in Data Base (31)	Log_{10} Number of Classes of Items in Data Base (32)	Log_{10} Number of Words in Tables and Constants not in Data Base (33)	Log_{10} Number of Input Message Types (34)	Log_{10} Number of Output Message Types (35)
50	4.70	0	1.50	2.70	-0	(0)	0	0
51	4.23	4.48	1.56	4.34	0.85	3.70	0.48	0.79
52	4.20	0	1.41	(5.30)	3.00	0	1.66	1.92
53	5.02	3.70	1.38	4.11	0.60	(2.48)	1.20	1.04
54	5.41	4.18	2.08	7.68	3.90	7.08	1.69	1.93
55	5.51	3.78	2.02	8.38	0.60	6.70	1.86	1.90
56	5.44	4.40	0.48	0	-0	0	0	0
57	3.88	0	1.00	3.60	0.70	(2.18)	0.60	0
58	4.20	3.95	1.23	(4.93)	(1.41)	(2.40)	0.60	0
59	4.23	5.12	0.48	0	-0	0	0	0
60	3.48	3.48	0.95	3.90	1.90	2.90	1.34	1.68
61	3.66	0	1.67	3.91	0.85	2.90	-0	0.48
62	2.85	2.02	0.48	2.08	0.85	1.83	-0	-0
63	3.43	2.66	-0	4.60	1.04	3.07	0.70	0.70
64	3.20	2.30	0.48	4.60	0.78	2.40	-0	0.30
65	3.18	2.40	0.48	6.18	0.90	2.70	0.48	0.30
66	3.65	3.28	-0	6.60	2.10	3.70	1.00	1.30
67	3.27	2.84	0.60	3.57	1.67	0	0	1.18
68	4.28	3.70	0.60	4.11	2.08	4.65	0.30	1.26
69	4.04	4.93	1.49	(3.58)	(0.85)	(3.26)	0.30	0.48
70	4.60	4.94	1.08	5.30	3.31	3.48	0.48	0.30
71	4.40	0	0.78	4.30	0.90	2.18	-0	-0
72	4.93	2.32	1.45	(4.50)	(1.00)	2.00	0.30	0.48
73	4.11	4.94	1.51	5.76	2.41	3.40	1.38	1.52
74	3.92	3.04	0.30	4.24	2.18	3.00	0	0

APPENDIX III (Cont'd)

Sample Number	Complexity of Program Design (36)	% Clerical Instructions (37)	% Math Instructions (38)	% Input/ Output Instructions (39)	% Logical Control Instructions (40)	% Self-Checking Fix Instructions (41)	% Information Storage and Retrieval (42)	% Data Acquisition and Display (43)	% Decision Making (45)
1	2	0.50	0	0.50	0	0	0.30	0	0
2	1	0.10	0	0	0.90	0	0.10	0	0
3	2	0.40	0.10	0.40	0.10	0	0.30	0.30	0
4	1	0.50	0	0.40	0.10	0	0.40	0.30	0.10
5	1	0.78	0.01	0.19	0.02	0	0	0	0
6	4	0.65	0	0.10	0	0.25	0.60	0	0.05
7	3	0.10	0	0.50	0.40	0	0.10	0.50	0
8	3	0.10	0	0.50	0.40	0	0.10	0.50	0
9	3	0.20	0	0.30	0.30	0.20	0.30	0.30	0.20
10	2	0.50	0.05	0.40	0.05	0	0.70	0	0.05
11	2	0.40	0.10	0.40	0.10	0	0.45	0	0
12	3	0.20	0.20	0.50	0.10	0	0.25	0.10	0.05
13	3	0.70	0	0.10	0.20	0	0.20	0	0.40
14	4	0.25	0	0.50	0.25	0	0	0.99	0
15	2	0.01	0.99	0	0	0	0	0.67	0
16	3	0.10	0	0.50	0.40	0	0.10	0.50	0
17	2	0.05	0	0.88	0.05	0.02	0.25	0.20	0
18	2	0.03	0.01	0.90	0.05	0.01	0.10	0.10	0.10
19	3	0.10	0	0.50	0.30	0.10	0.30	0.50	0.10
20	3	0.10	0.05	0.10	0.60	0.05	0	0.70	0
21	3	0.40	0	0.40	0.10	0.10	0.50	0.20	0
22	4	0.45	0.05	0.25	0.25	0	0.25	0.25	0.05
23	3	0.20	0.05	0.70	0.03	0.02	0.10	0	0.20
24	4	0.99	0	0	0	0	0	0	0

APPENDIX III (Cont'd)

Sample Number	Complexity of Program Design (36)	% Clerical Instructions (37)	% Math Instructions (38)	% Input/ Output Instructions (39)	% Logical Control Instructions (40)	% Self-Checking Fix Instructions (41)	% Information Storage and Retrieval (42)	% Data Acquisition and Display (43)	% Decision Making (45)
25	3	0.30	0.10	0.40	0.20	0	0	0.60	0.40
26	3	0.20	0.05	0.50	0.25	0	0.50	0.20	0
27	3	0	0.65	0.20	0.15	0	0	0	0.80
28	3	.30	0	0	.70	0	0	0	0
29	2	0.40	0.10	0.30	0.20	0	0.20	0	0.05
30	3	0.05	0.05	0.50	0.20	0.20	0.05	0.60	0
31	3	0.10	0.01	0.78	0.10	0.01	0.60	0.15	0
32	3	0.20	0.40	0.20	0.05	0.15	0.10	0.20	0.30
33	3	0.17	0.03	0.25	0.35	0	0.05	0.05	0.05
34	2	0.30	0	0.60	0.10	0	0	0	0
35	4	0.08	0.02	0.60	0.30	0	0.20	0.80	0
36	1	0.64	0.25	0.10	0.01	0	0.50	0	0
37	3	0.69	0.03	0.20	0.05	0.03	0.40	0	0.05
38	4	0.10	0	0.60	0.30	0	0	0	0
39	4	0.17	0.18	0.45	0.16	0.04	0.10	0.56	0
40	2	0.25	0	0.60	0.15	0	0	0	0.20
41	4	0.40	0.10	0.35	0.10	0.05	0.20	0.30	0.05
42	4	0.10	0.30	0.10	0.30	0.20	0	0.20	0.40
43	2	0.20	0.20	0.30	0.30	0	0.60	0	0.20
44	4	0.28	0.20	0.20	0.22	0.10	0.30	0.25	0.10
45	4	0.10	0.05	0.75	0.10	0	0	0	0
46	4	0.60	0.15	0.05	0.20	0	0.99	0	0
47	4	0.15	0.15	0.15	0.45	0.10	0.10	0.05	0.05
48	4	0.10	0.40	0.10	0.10	0	0.20	0.40	0.20
49	4	0.20	0.28	0.20	0.22	0.10	0.25	0.20	0.15

APPENDIX III (Cont'd)

Sample Number	Complexity of Program Design (36)	% Clerical Instructions (37)	% Math Instructions (38)	% Input/ Output Instructions (39)	% Logical Control Instructions (40)	% Self-Checking Fix Instructions (41)	% Information Storage and Retrieval (42)	% Data Acquisition and Display (43)	% Decision Making (45)
50	2	0.50	0	0.50	0	0	0.20	0	0
51	4	0.18	0.30	0.20	0.22	0.10	0.20	0.20	0.20
52	5	0.02	0.01	0.20	0.74	0.03	0	0	0.15
53	3	0.02	0.05	0.75	0.04	0.02	0.20	0.75	0
54	5	0.25	0.05	0.10	0.50	0.10	0.30	0.10	0
55	5	0	0.15	0.50	0.35	0	0.15	0	0
56	3	0.05	0.05	0.20	0.70	0	0.16	0.17	0.17
57	3	0.30	0.30	0.20	0.20	0	0.20	0	0.10
58	2	0.10	0.20	0.55	0.15	0	0	0	0.20
59	3	0.80	0	0.20	0	0	0.30	0	0
60	5	0.10	0.20	0.40	0.20	0.10	0.40	0	0
61	2	0.13	0.02	0.24	0.10	0.41	0.20	0	0.10
62	2	0.15	0.10	0.45	0.30	0	0.30	0.30	0
63	2	0.15	0	0.80	0.05	0	0.50	0	0
64	3	0.85	0	0.10	0.05	0	0.80	0.02	0.05
65	3	0.80	0	0.10	0.10	0	0.70	0.10	0.04
66	3	0.10	0.05	0.25	0.50	0.10	0.50	0.05	0.15
67	3	0.67	0.03	0.10	0.05	0.15	0.80	0	0
68	3	0.72	0.09	0.07	0.07	0.02	0.30	0.02	0.05
69	4	0.09	0.38	0.49	0.03	0.01	0	0.95	0
70	3	0.70	0.05	0.10	0.10	0.05	0.70	0.15	0.03
71	5	0.90	0	0.05	0.05	0	0.99	0	0
72	4	0.11	0.37	0.48	0.03	0.01	0	0.95	0
73	2	0.15	0.25	0.50	0.04	0	0	0.67	0
74	1	0	0.05	0.10	0.84	0.01	0.15	0.15	0.55

APPENDIX III (Cont'd)

Sample Number	% Generation (47)	Insufficient Memory Capacity (50)	Insufficient Input/ Output Capacity (51)	Stringent Timing Requirements (52)	First Programming Effort on Computer (64)	Log_{10} Average Turnaround Time (65)	Log_{10} Number of ADP Components Developed with Program (66)	Number of Agencies Concurring in Design (99)
1	0	0	0	0	0	1.78	0	3
2	0	0	0	0	0	1.86	0	2
3	0	0	0	0	0	1.56	0	2
4	0	0	0	0	1	1.68	0	1
5	0.20	0	0	0	0	0.60	0	2
6	0.10	1	1	0	1	1.38	0	0
7	0	1	1	1	0	0	0	3
8	0	1	1	1	0	0	0	3
9	0	1	1	1	0	0	0	3
10	0	1	1	0	0	0.60	0	1
11	0	0	0	0	0	1.08	0	1
12	0.05	0	0	0	0	1.56	0	2
13	0.20	0	0	1	0	1.38	0	0
14	0	1	1	1	0	1.56	0	2
15	0.33	0	0	0	0	1.68	0	3
16	0	1	1	1	0	0	0	3
17	0.30	0	0	0	0	1.86	0	2
18	0.40	0	0	0	0	1.86	0	2
19	0	1	1	1	0	0	0	3
20	0	0	0	1	1	0	0.78	1
21	0	1	0	1	0	0	0	3
22	0.05	1	1	1	0	1.38	0	0
23	0.05	0	0	0	0	1.86	0	2
24	0	0	0	0	0	1.56	0	1

APPENDIX III (Cont'd)

Sample Number	% Generation (47)	Insufficient Memory Capacity (50)	Insufficient Input/ Output Capacity (51)	Stringent Timing Requirements (52)	First Programming Effort on Computer (64)	$\log_{10}$ Average Turnaround Time (65)	$\log_{10}$ Number of ADP Components Developed with Program (66)	Number of Agencies Concurring in Design (99)
25	0	1	1	1	0	0	-0	4
26	0	1	1	0	0	0	0	3
27	0.20	0	0	0	0	1.38	0	3
28	0	1	0	0	0	1.20	0	2
29	0.50	0	0	0	1	1.38	0	0
30	0	1	1	1	0	0	0	3
31	0	1	1	0	0	0	0	3
32	0.30	1	0	1	0	1.56	0	5
33	0.05	1	1	1	0	-0	1.00	6
34	0.30	1	0	0	1	-0	1.53	4
35	0	1	0	0	1	0	0.48	1
36	0	1	1	1	1	0.85	0	1
37	0.25	0	0	0	0	1.38	0	1
38	0	0	1	1	0	1.68	-0	1
39	0	1	0	0	1	1.38	1.20	2
40	0.20	1	0	1	0	-0	0	4
41	0.15	1	0	1	0	1.48	0	3
42	0.30	0	1	0	0	2.00	-0	1
43	0.20	0	0	0	1	2.00	0	0
44	0.15	0	1	1	0	1.86	0.60	0
45	0.50	1	1	1	0	1.56	0	2
46	0	1	1	1	1	1.56	0.78	2
47	0.50	1	1	0	1	0.78	0	4
48	0	1	1	0	0	1.68	0.30	1
49	0.15	0	1	1	0	1.68	0	0

APPENDIX III (Cont'd)

Sample Number	% Generation (47)	Insufficient Memory Capacity (50)	Insufficient Input/ Output Capacity (51)	Stringent Timing Requirements (52)	First Programming Effort on Computer (64)	Log_{10} Average Turnaround Time (65)	Log_{10} Number of ADP Components Developed with Program (66)	Number of Agencies Concurring in Design (99)
50	0.30	0	0	0	1	-0	-0	0
51	0.15	0	1	1	0	1.68	0	0
52	0.05	1	0	1	1	1.38	1.30	2
53	0.05	1	0	1	1	1.38	0.70	4
54	0.30	1	1	1	0	1.38	-0	8
55	0.70	1	1	0	0	1.68	0	2
56	0.17	0	1	1	1	2.00	0	3
57	0.20	1	0	0	1	-0	0	5
58	0.30	1	0	1	0	-0	0	5
59	0.20	0	1	0	0	1.56	0	5
60	0.20	0	0	0	0	1.38	0	3
61	0.15	0	0	1	0	1.38	0	4
62	0.20	0	0	1	0	1.68	0	4
63	0	1	1	0	0	1.48	0	1
64	0.03	0	0	0	0	-0	0	2
65	0.06	0	0	0	0	-0	0	2
66	0.20	0	1	0	0	1.56	0	1
67	0.10	0	1	0	1	1.68	-0	3
68	0.41	1	1	1	0	1.08	0	4
69	0.05	1	0	1	0	1.68	0.30	3
70	0	0	0	0	0	1.38	0	1
71	0	0	0	0	1	1.86	0	2
72	0.05	1	1	1	1	1.68	0.30	3
73	0.33	0	0	1	0	1.68	0	3
74	0.05	0	0	0	1	1.38	0	1

APPENDIX III (Cont'd)

Sample Number	Estimated Customer Experience (100)	Computer Operated by Agency Other Than Developer (101)	Open/Closed Shop (102)	Time Sharing (103)	Program Developed Away from Operational Location (104)	Computer at Operation Site Different Than at Development Site (105)	Program Developed at More Than One Location (106)
1	3	0	3	0	1	0	0
2	2	1	1	0	1	0	0
3	3	1	1	0	1	0	0
4	3	1	1	0	0	0	0
5	2	1	3	0	0	0	0
6	1	1	1	0	0	0	0
7	3	1	3	0	1	0	1
8	3	1	3	0	1	0	0
9	3	1	3	0	1	0	1
10	2	1	3	0	1	0	1
11	2	0	2	1	0	0	0
12	3	1	1	0	1	0	0
13	1	1	1	0	0	0	0
14	3	1	1	0	1	0	1
15	3	0	3	0	1	0	0
16	3	1	3	0	1	0	1
17	2	1	1	0	1	0	0
18	2	1	1	0	1	0	0
19	3	1	3	0	1	0	1
20	3	0	3	0	0	0	0
21	3	1	3	0	1	0	1
22	3	0	3	0	0	0	0
23	2	1	1	0	1	0	0
24	3	1	1	0	1	0	0

APPENDIX III (Cont'd)

Sample Number	Estimated Customer Experience (100)	Computer Operated by Agency Other Than Developer (101)	Open/Closed Shop (102)	Time Sharing (103)	Program Developed Away from Operational Location (104)	Computer at Operation Site Different Than at Development Site (105)	Program Developed at More Than One Location (106)
25	3	0	3	0	1	0	0
26	3	1	3	0	1	0	1
27	2	1	1	0	1	0	1
28	3	1	3	0	1	0	1
29	2	1	3	0	0	0	0
30	3	1	3	0	1	0	1
31	3	1	3	0	1	0	1
32	2	1	3	0	1	0	0
33	2	1	3	0	0	0	0
34	2	0	3	0	1	1	0
35	3	0	0	1	0	0	0
36	2	1	2	0	1	0	1
37	2	1	3	0	1	0	1
38	2	1	2	0	0	0	1
39	2	1	2	0	1	0	1
40	2	0	3	0	1	1	1
41	3	0	3	0	1	0	0
42	2	1	2	0	1	0	1
43	2	1	2	0	1	0	1
44	3	1	1	0	1	0	1
45	3	1	1	0	1	0	0
46	1	1	2	0	1	0	0
47	3	0	1	0	0	0	0
48	2	1	3	0	1	0	0
49	3	1	1	0	1	0	1

APPENDIX III (Cont'd)

Sample Number	Estimated Customer Experience (100)	Computer Operated by Agency Other Than Developer (101)	Open/Closed Shop (102)	Time Sharing (103)	Program Developed Away from Operational Location (104)	Computer at Operation Site Different Than at Development Site (105)	Program Developed at More Than One Location (106)
50	2	0	3	0	1	1	1
51	3	1	1	0	1	0	1
52	1	1	2	0	1	0	1
53	2	1	1	0	1	0	1
54	2	1	2	0	1	0	0
55	2	1	2	0	1	0	1
56	2	1	2	0	1	0	0
57	2	0	3	0	1	1	0
58	2	0	3	0	1	1	1
59	3	1	1	0	0	0	0
60	2	1	1	0	1	0	0
61	3	1	1	0	1	0	0
62	3	1	1	0	1	1	0
63	2	1	3	0	0	0	0
64	2	1	2	1	0	0	0
65	2	1	2	1	0	0	0
66	2	1	3	0	0	0	0
67	2	1	2	0	0	0	0
68	3	1	1	0	1	0	0
69	2	1	3	0	0	0	0
70	3	0	1	0	0	0	0
71	2	1	2	0	1	1	1
72	2	1	3	0	0	0	0
73	2	1	3	0	0	0	0
74	3	0	2	0	0	0	0

APPENDIX III (Cont'd)

Sample Number	% Transform-Reformatting (46)	% Error Rate (machine) (109)	$\log_{10}$ Production Rate (machine) (111)	% Operational Discards (115)	$\log_{10}$ Travel Rate (117)	$\log_{10}$ Pages Documentation (118)	$\log_{10}$ Documentation Rate (machine) (119)	$\log_{10}$ Months Elapsed (121)	$\log_{10}$ Computer Usage Rate (machine) (123)
1	0.70	0	2.42	0	0	2.03	-0.40	0.60	-1.17
2	0	0	0.70	0	-0.63	1.57	-0.60	0.70	-1.88
3	0.40	0	1.88	0	0	2.21	0.03	0.48	-1.48
4	0.10	0.14	2.01	0	0	1.23	-1.26	0.60	-0.89
5	0.80	0	2.05	0	0	1.88	-0.65	0.60	-1.45
6	0.20	0.12	2.83	0.01	0	1.97	-1.33	1.00	-2.13
7	0	0.05	2.40	0.05	0.40	1.90	-1.27	0.78	-1.88
8	0	0.04	2.58	0.05	0.22	2.08	-1.28	0.78	-2.06
9	0	0.06	2.40	0.05	0.40	1.97	-1.20	0.78	-1.88
10	0.20	0.08	2.54	0.06	0	1.70	-1.62	0.70	-2.01
11	0.45	0.04	2.56	0.16	0	1.40	-1.94	0.30	-3.04
12	0.50	0.04	2.30	0.02	0	2.42	-0.66	0.60	-1.50
13	0	0.12	1.92	0.13	0	0	0	0.48	-0.85
14	0	0.28	2.24	0	0	2.00	-1.02	1.04	-1.62
15	0	0	2.06	0.03	0	2.33	-0.50	0.90	-1.33
16	0	0.06	2.33	.05	0.40	2.18	-0.99	0.78	-1.88
17	0.25	0.04	1.95	0	-1.88	2.05	-1.37	0.70	-1.99
18	0.20	0	1.95	0	-1.89	1.94	-1.49	0.70	-2.20
19	0	0.02	2.68	0.04	-0.01	2.33	-1.25	0.78	-2.28
20	0.10	0.07	2.57	0.40	0.30	1.40	-2.08	0.60	-1.27
21	0.20	0.09	2.54	0.01	1.24	2.44	-1.10	0.78	-1.37
22	0.25	(0.06)	2.71	(0.44)	0	1.60	-2.19	1.20	-1.49
23	0.60	0.03	2.37	0.06	-2.30	2.26	-1.59	0.70	-2.41
24	0	0.22	2.51	0.28	0	2.30	-1.35	0.60	-1.98

Sample Number	% Transform-Reformatting (46)	% Error Rate (machine) (109)	Log_{10} Production Rate (machine) (111)	% Operational Discards (115)	Log_{10} Travel Rate (117)	Log_{10} Pages Documentation (118)	Log_{10} Documentation Rate (machine) (119)	Log_{10} Months Elapsed (121)	Log_{10} Computer Usage Rate (machine) (123)
25	0	0.04	2.80	0	0.10	2.43	-1.57	0.60	-1.92
26	0.25	0.03	2.49	0.05	0.78	2.25	-1.44	1.04	-1.82
27	0.20	0.08	2.50	0.07	0.68	1.88	-1.90	0.60	-2.00
28	0.99	(0.09)	2.81	0.16	0.29	2.41	-1.68	0.95	-2.60
29	0.15	0	3.65	0.01	-1.13	2.36	-2.59	1.23	-2.26
30	0.05	0.03	2.40	0.05	1.01	2.08	-1.69	1.04	-1.60
31	0.20	0.13	2.33	0.01	1.05	2.55	-1.19	1.04	-1.56
32	0.05	0.05	2.48	0	-0.30	2.40	-1.56	0.60	-1.56
33	0.05	0.08	2.52	0	0	2.95	-1.22	0.60	-2.05
34	0.60	0.05	2.30	0	0.06	(3.32)	(-0.70)	0.95	-1.61
35	0	0.08	2.77	0	0	2.79	-1.75	1.11	-1.80
36	0	0.06	2.50	0.16	0	2.48	-1.82	0.90	-1.50
37	0.30	0.24	2.61	0.08	0	3.51	-1.00	0.70	-1.73
38	0	0.09	2.57	0.18	-0.91	(3.20)	(-1.35)	1.40	-2.08
39	0.18	0.07	2.47	0.18	-0.81	2.81	-1.70	0.70	-1.73
40	0.45	0.02	2.20	0.04	-0.10	(3.00)	(-1.28)	1.26	-1.63
41	0.25	0.03	2.09	0.10	0.30	4.16	-0.04	1.23	-1.26
42	0	0.14	2.87	0.19	0	4.42	-0.60	1.28	-2.91
43	0	0.15	2.63	0	0.23	2.58	-2.31	1.15	-2.60
44	0.10	0.12	2.23	0.23	0.38	3.28	-1.21	1.04	-1.73
45	0.50	0.29	2.12	0.33	0.08	3.30	-1.10	1.26	-1.25
46	0	0.01	2.45	0.04	0	3.62	-1.13	1.18	-2.25
47	0.25	0.28	2.64	0.03	0.05	3.89	-1.07	1.38	-1.96
48	0.20	0.15	2.73	0.22	-1.09	3.03	-2.06	1.30	-2.16
49	0.10	0.05	2.10	0.15	0.15	3.40	-1.11	1.11	-1.39

Sample Number	% Transform-Reformatting (46)	% Error Rate (machine) (109)	Log_{10} Production Rate (machine) (111)	% Operational Discards (115)	Log_{10} Travel Rate (117)	Log_{10} Pages Documentation (118)	Log_{10} Documentation Rate (machine) (119)	Log_{10} Months Elapsed (121)	Log_{10} Computer Usage Rate (machine) (123)
50	0.50	0.03	2.29	0.16	0.09	3.20	-1.51	1.32	-1.50
51	0.10	0.05	1.77	0.04	0.54	3.48	-0.77	1.56	-1.20
52	0	0.43	2.33	0.63	0.15	2.70	-2.37	1.36	-2.06
53	0	0.14	2.78	0.36	-0.81	3.40	-2.16	1.26	-2.20
54	0.20	(0.06)	2.50	(0.36)	0.07	4.15	-1.51	1.23	-1.74
55	0	(0.05)	2.55	(0.44)	0.02	4.30	-1.42	1.45	-1.76
56	0.16	(0.04)	2.46	(0.28)	0	3.18	-2.50	1.75	-1.82
57	0.30	0.06	2.17	0.13	0.19	2.60	-1.29	0.95	-1.45
58	0.40	0.02	2.15	0.07	-0.05	2.95	-1.28	1.26	-1.58
59	0.50	0.08	2.80	0.02	0	2.64	-1.59	0.48	-1.88
60	0.20	0.10	1.78	0.10	0	2.31	-1.17	0.48	-1.48
61	0.05	0.05	3.18	0.17	0	0.85	-2.82	0.78	-2.19
62	0.10	0.01	2.28	0.06	0	1.90	-0.93	0.70	-1.66
63	0.50	0.08	2.83	0	0	1.68	-1.82	0.48	-2.66
64	0.05	0.16	2.60	0.56	0	1.54	-1.71	0.60	-1.49
65	0.05	0.20	2.57	.29	0	1.54	-1.69	0.60	-1.83
66	0	0.09	3.20	0	-0.94	1.28	-2.80	0.60	-2.82
67	0.10	0.02	2.00	0.20	0.38	1.43	-1.84	0.48	-1.65
68	0.14	0.21	2.71	0	-0.54	2.90	-1.46	0.90	-2.52
69	0	(0.08)	2.38	(0.01)	0.42	2.65	-2.33	1.11	-2.55
70	0.07	0.03	2.92	0.14	-0.71	2.63	-2.57	1.55	-1.86
71	0	0.09	1.77	2.20	0	3.65	-0.76	1.34	-1.77
72	0	(0.05)	2.39	(0.06)	0.10	3.74	-1.44	1.38	-1.90
73	0	0	2.20	0	0.48	3.44	-1.58	1.40	-2.62
74	0.05	0.10	2.75	0.05	-0.48	1.40	-1.06	0.85	-1.97

APPENDIX III (Cont'd)

Sample Number	Log_{10} Data Turnover Index (127)	% Senior Programmers (128)	% Programmers Participating in Design (143)	% Programmers for Duration (144)	Log_{10} Staff Application Experience (146)	Log_{10} Staff Development Computer Experience (147)	Log_{10} Staff Development Language Experience (148)	Costliness Factor (154)
1	2.57	0.67	0.17	0	0.56	0.56	0.56	-1.48
2	0	0.33	1.00	0	0	-0	-0	-1.24
3	2.28	0	1.00	1.00	0.30	0.30	0.30	-1.52
4	3.45	0	0.67	1.00	-0.48	-0.48	-0.48	-1.32
5	4.41	0	0.50	0.50	0	0.58	0.48	-1.32
6	3.07	0	1.00	1.00	0	0.70	0.70	-1.04
7	0.70	1.00	1.00	1.00	0.60	0.60	0.60	-0.94
8	0.70	1.00	1.00	1.00	0.60	0.60	0.60	-0.86
9	0.70	1.00	1.00	1.00	0.60	0.60	0.60	-0.93
10	5.22	0.83	0.50	1.00	0.22	0.50	0.50	-0.93
11	0.70	1.00	1.00	0.50	0	0.30	0.48	-1.18
12	2.51	0.50	1.00	1.00	0.30	0.70	0.70	-0.83
13	2.90	0	1.00	1.00	0.30	0.48	0.48	-1.28
14	2.70	0.07	0.21	0	0	0.29	0.29	-0.95
15	1.82	0.56	0.25	0	0.52	0.52	0.52	-0.94
16	0.70	1.00	1.00	0.50	0.60	0.60	0.60	-0.87
17	3.18	0.33	1.00	0	0	-0	-0	-0.53
18	3.54	0.33	1.00	0	0	-0	-0	-0.59
19	2.00	1.00	1.00	0.50	0.60	0.60	0.60	-0.70
20	2.62	1.00	1.00	0.50	0	0	0	-0.74
21	1.30	0.33	0.67	0.67	0.30	0.30	0.30	-0.11
22	0.99	1.00	1.00	1.00	0.30	0.90	0.48	-0.48
23	2.76	0.33	1.00	0	0	-0	-0	-0.37
24	0	0.25	0.43	0.29	0.35	0.68	0.68	-0.51

APPENDIX III (Cont'd)

Sample Number	$\log_{10}$ Data Turnover Index (127)	% Senior Programmers (128)	% Programmers Participating in Design (143)	% Programmers for Duration (144)	$\log_{10}$ Staff Application Experience (146)	$\log_{10}$ Staff Development Computer Experience (147)	$\log_{10}$ Staff Development Language Experience (148)	Costliness Factor (154)
25	3.43	0.17	0.33	0.17	-0	0.07	0.07	-0.09
26	0.70	0.50	1.00	0.50	0.40	0.40	0.40	-0.16
27	2.52	0.50	0.75	0.75	0.48	0.24	0.24	-0.19
28	0	1.00	1.00	1.00	0.48	0.78	0.48	-0.16
29	0	0.67	0.20	0.20	0.48	0	0	0.60
30	0.70	0.33	0.67	0.67	0.30	0.30	0.30	0.06
31	1.00	0.33	0.67	0.67	0.30	0.30	0.30	0.13
32	0.05	0.43	0.33	0.50	0.41	0.41	0.41	0.05
33	0.70	0.67	0.88	1.00	-0	-0	0.37	0.16
34	2.82	0.25	1.00	0.67	-0.43	0	0	0.34
35	5.00	1.00	0.80	0.60	0	0.15	0.60	0.33
36	2.92	0.38	0.50	0.20	-0	0.30	-0	0.19
37	0.85	0.45	0.70	0.80	-0.34	0.16	0.21	0.46
38	2.02	0.33	0.75	0.13	0	0	0	0.51
39	3.90	0.53	0.25	0.63	0.55	0	0	0.77
40	3.60	0.38	1.00	1.00	-0	-0	-0	0.60
41	2.65	0.57	0.34	0.07	0.41	0.50	0.50	0.92
42	3.24	0.50	0.40	0.67	0	-0	0.43	0.58
43	0	0.33	0	0.67	0	0	0.53	0.95
44	3.52	0.32	0.46	0.75	-0.62	0.18	0.02	0.96
45	2.91	0.33	0.33	0.22	0.38	0.38	0.38	1.03
46	4.55	0.10	0.10	0.50	0	0	0	0.61
47	2.43	0.22	0.82	0	0.44	0.48	0.48	1.24
48	4.20	0	0.15	0.05	0.43	-0	0.26	0.98
49	3.37	0.44	0.44	0.63	0.55	0.55	0.55	1.09

APPENDIX III (Cont'd)

Sample Number	$\log_{10}$ Data Turnover Index (127)	% Senior Programmers (128)	% Programmers Participating in Design (143)	% Programmers for Duration (144)	$\log_{10}$ Staff Application Experience (146)	$\log_{10}$ Staff Development Computer Experience (147)	$\log_{10}$ Staff Development Languages Experience (148)	Costliness Factor (154)
50	0	0.42	0.71	0.43	0	0	0	1.20
51	3.39	0.50	0.47	0.47	0.68	0.68	0.68	1.08
52	3.18	0.30	0.25	0.15	-0	-0	0.05	1.54
53	2.68	0.13	0.16	0.66	0.53	0.48	0.48	1.45
54	5.55	0.14	0.03	0.29	0.32	0.32	0.32	2.21
55	6.19	0.16	0.13	0.49	0.25	0.20	0.20	2.33
56	-0	0.80	0.58	0.35	0.30	-0	0.30	1.49
57	3.00	0.25	1.00	0.83	-0.43	0	0	0.19
58	4.33	0.38	1.00	1.00	-0	-0	-0	0.56
59	-0	0.33	0.90	0.50	0.37	0.52	0.30	-0.01
60	2.06	0.57	0.33	0.17	0.20	0.20	0.20	-0.26
61	3.31	0	1.00	1.00	0	0.48	0.48	-1.05
62	1.78	0	1.00	1.00	-0	0.48	-0	-1.19
63	3.60	1.00	1.00	1.00	-0	-0	-0	-1.07
64	4.12	1.00	1.00	1.00	0.30	0.48	0.48	-0.94
65	5.48	1.00	1.00	1.00	0.30	0.48	0.48	-1.05
66	5.12	1.00	1.00	1.00	0	-0	0.30	-0.98
67	2.40	1.00	1.00	1.00	0	-0	-0	-0.59
68	2.81	1.00	0.75	0.50	0	0.48	0.48	0.06
69	3.88	0.67	0.30	0.30	0.45	0.37	0.36	0.77
70	4.60	0.75	0.50	0.50	0	0	0	0.61
71	4.00	0.38	0.40	0.30	0	-0	0.35	0.60
72	3.80	0.57	0.30	0.30	0.28	0.39	0.46	2.10
73	4.00	0.70	0.90	0.80	0.46	0.53	0.49	1.10
74	0	0.50	1.00	1.00	0	0	0.40	-0.43

APPENDIX III (Cont'd)

Sample Number	Uniqueness Index (149)	Job Difficulty Index (150)	Development Environment Index (151)	Type of Job Index (152)	Staffing Index (153)
1	0	4.59	-3.57	-1.41	0.67
2	0	0	-5.40	-0.10	0.33
3	0	2.99	-6.40	-0.92	0
4	2.90	2.99	-5.27	-0.63	0
5	0	3.50	-3.70	-0.94	0
6	2.90	3.20	-4.40	-0.47	0
7	2.30	4.67	-6.40	-0.10	1.00
8	1.00	5.55	-6.40	-0.10	1.00
9	2.30	6.17	-6.40	-0.20	1.00
10	1.30	5.37	-3.70	-0.76	0.83
11	0	0.70	-6.40	-0.99	1.00
12	0	5.47	-6.40	-0.63	0.50
13	2.20	2.87	-4.40	0.18	0
14	2.30	10.10	-3.73	-0.25	0.07
15	0	3.31	-3.85	1.44	0.56
16	3.50	6.80	-6.40	-0.10	1.00
17	0	4.97	-5.40	0.95	0.33
18	0	1.30	-5.40	1.47	0.33
19	3.50	5.99	-6.40	-0.10	1.00
20	3.90	5.62	-6.40	-0.23	1.00
21	2.30	7.17	-5.27	-0.66	0.33
22	2.20	6.73	-6.40	-0.56	1.00
23	0	1.70	-5.40	-0.76	0.33
24	1.20	3.49	-4.46	-0.99	0.25

APPENDIX III (Cont'd)

Sample Number	Uniqueness Index (149)	Job Difficulty Index (150)	Development Environment Index (151)	Type of Job Index (152)	Staffing Index (153)
25	2.20	6.87	-4.13	-0.82	0.17
26	1.30	9.09	-6.40	-0.53	0.50
27	2.50	5.39	-4.55	0.88	0.50
28	1.30	10.06	-6.40	-1.59	1.00
29	2.90	3.71	-2.68	1.61	0.67
30	2.30	6.35	-5.27	-0.12	0.33
31	1.30	9.36	-5.27	-0.36	0.33
32	2.20	7.39	-3.13	1.06	0.43
33	2.20	7.51	-4.98	-0.02	0.67
34	2.90	6.51	-6.40	0.24	0.25
35	2.90	8.43	-5.72	-0.08	1.00
36	4.00	5.82	-3.70	-0.64	0.38
37	2.50	0	-4.38	0.02	0.45
38	3.50	4.10	-4.55	-0.10	0.33
39	3.00	7.63	-2.85	-0.40	0.53
40	2.30	7.17	-5.89	0.05	0.38
41	1.00	8.80	-4.17	-0.07	0.57
42	2.50	7.85	-3.36	1.22	0.50
43	3.00	1.00	-2.00	0.68	0.33
44	3.50	8.63	-4.58	0.25	0.32
45	1.00	10.65	-4.13	1.45	0.33
46	2.70	8.03	-1.34	-0.60	0.10
47	2.90	11.34	-5.78	1.73	0.22
48	0	8.63	-2.51	-0.36	0
49	3.50	8.63	-4.49	0.33	0.44

APPENDIX III (Cont'd)

Sample Number	Uniqueness Index (199)	Job Difficulty Index (150)	Development Environment Index (151)	Type of Job Index (152)	Staffing Index (153)
50	3.00	7.46	-4.43	0.17	0.42
51	3.50	8.63	-4.61	0.35	0.50
52	5.20	10.07	-1.85	0.20	0.30
53	5.20	7.50	-2.54	0.20	0.13
54	2.20	14.32	-2.09	0.81	0.14
55	2.50	10.71	-2.44	3.08	0.16
56	3.90	2.39	-3.98	0.49	0.80
57	2.90	5.70	-5.40	0.19	0.25
58	3.50	7.57	-5.89	0.70	0.38
59	1.20	2.39	-6.06	-0.57	0.33
60	0	6.67	-3.13	0.52	0.57
61	1.00	9.21	-6.40	0.47	0
62	1.00	3.23	-6.40	0.60	0
63	1.20	1.04	-5.40	-0.80	1.00
64	1.20	3.16	-5.40	-0.78	1.00
65	1.20	3.29	-5.40	-0.60	1.00
66	1.20	2.10	- 6.40	0.78	1.00
67	2.90	4.68	-5.40	-0.36	1.00
68	2.20	5.09	-5.55	0.90	1.00
69	2.20	8.30	-3.02	0.13	0.67
70	1.20	8.71	-4.70	-0.79	0.75
71	4.20	4.79	-3.36	-0.90	0.38
72	3.90	8.24	-3.02	0.11	0.57
73	2.20	9.93	-5.06	1.30	0.70
74	2.90	3.68	-6.40	0.16	0.50

(Page 152 Blank)

APPENDIX IV

VALIDITY OF SELECTED PREDICTORS

This Appendix contains the correlations, (i.e., validities), of selected predictors with the major dependent variables. The dependent variables are identified in column headings; the predictors are noted both by title and by the identifying number used throughout the analysis.

A more detailed identification of all variables is contained in Appendix II.

		Log_{10} Total Man Months (1)	Log_{10} Total Computer Hours (9)	Log_{10} New Instructions (20)	Log_{10} Months Elapsed (121)	Costliness Factor (154)
12	Innovation in system	23	35	34	19	29
13	Participation by programming organization	11	13	12	14	10
14	How well were plans known and documented	- 6	- 13	- 14	3	- 4
15	Log_{10} number users supplying input and requesting output	17	9	12	6	14
16	Log_{10} number ADP centers in system	- 17	- 10	- 22	- 7	-17
17	System complexity	40	50	44	28	45
30	Log_{10} number of subprograms	46	57	44	55	57
31	Log_{10} number of words in Data Base	47	37	41	34	41
32	Log_{10} number of classes of items in Data Base	20	17	24	14	22
33	Log_{10} number of words in tables and constants not in Data Base	21	24	28	18	27
34	Log_{10} number of input message types	24	24	24	21	29
35	Log_{10} number of output message types	21	16	14	12	24
36	Complexity of program design	48	49	47	39	51
37	Percent clerical instructions	- 25	- 9	- 16	- 23	-23
38	Percent math instructions	19	17	13	14	22
39	Percent I/O instructions	0	- 6	- 2	0	2
40	Percent logical control instructions	17	3	9	17	8
41	Percent self-check-fix instructions	- 8	- 4	1	- 1	4
42	Percent information storage and retrieval	- 15	- 10	- 5	- 13	-16
43	Percent data acquisition and display	0	7	- 1	13	8

		Log_{10} Total Man Months (1)	Log_{10} Total Computer Hours (9)	Log_{10} New Instructions (20)	Log_{10} Months Elapsed (121)	Costliness Factor (154)
45	Percent decision making	5	2	6	- 1	1
46	Percent transformation-reformatting	- 15	- 14	- 12	- 18	- 13
47	Percent generation	36	35	35	27	29
50	Insufficient memory capacity	19	22	25	23	27
51	Insufficient I/O capability	16	13	24	22	18
52	Stringent timing requirements	21	27	16	27	24
64	First programming effort on computer	30	35	32	27	30
65	Log_{10} average turnaround time	23	9	12	15	17
66	Log_{10} number ADP components developed with program	28	26	21	8	28
99	Number of agencies concurring in design	10	10	9	- 6	13
100	Estimate customer experience	- 29	- 12	- 19	- 7	- 21
101	Computer operated by agency other than developer	0	- 13	- 4	- 12	- 2
102	Open-closed shop	- 7	1	2	4	3
103	Time-sharing	- 18	- 16	- 9	- 20	- 18
104	Program developed away from operational site	24	9	9	12	15
105	Computer at operation site different than development site	14	13	9	18	11
106	Program developed at more than one location	27	20	24	29	27
127	Log_{10} data turnover index	22	25	21	20	24
22	Log_{10} reused machine instructions	11	21	18	3	20
143	Percent programmers participating in design	- 45	- 51	- 36	- 39	- 49

		Log_{10} Total Man Months (1)	Log_{10} Total Computer Hours (9)	Log_{10} New Instructions (20)	Log_{10} Months Elapsed (121)	Costliness Factor (154)
144	Percent programmers for duration of project	- 33	- 20	- 17	- 28	- 28
146	Log_{10} staff application experience	- 3	4	1	9	5
147	Log_{10} staff development computer experience	- 28	- 18	- 20	- 8	- 19
148	Log_{10} staff development language experience	- 20	- 15	- 7	- 3	- 14
128	Percent senior programmers	- 21	- 20	- 5	- 14	- 18
10	Log_{10} number of trips	62	61	56	55	76
109	Percent error rate (machine)	18	19	16	12	19
111	Log_{10} production rate (machine)	- 8	14	39	9	10
115	Percent operational discards	27	27	23	25	23
117	Log_{10} travel rate	- 7	8	- 6	5	4
118	Log_{10} pages documented	81	72	74	69	84
119	Log_{10} documentation rate (machine)	- 12	- 18	- 40	- 11	- 16
123	Log_{10} computer usage rate (machine)	- 5	25	- 25	3	- 2
149	Uniqueness index	49	56	52	48	52
150	Job difficulty index	48	57	47	55	59
151	Development environment index	- 52	- 51	- 40	- 38	- 53
152	Type of job index	44	37	39	36	45
153	Staffing index	- 21	- 20	- 5	- 14	- 18

APPENDIX V

A LIST OF MAJOR DEPENDENT AND INDEPENDENT VARIABLES

This Appendix lists the dependent and independent variables, and their identifying numbers, that remained after preliminary statistical analysis to remove redundant and spurious relationships. For example, certain variables were eliminated from further consideration after correlation analysis indicated that they were weakly related to resource-costs and would contribute little to prediction.

ID No.	Title
10	Log_{10} number of trips
11	Log_{10} average distance/trip
12	Innovation in system
13	Participation by programming organization in requirement and/or operational design
14	How well were operational requirements known and documented
15	Log_{10} number of users supplying input to and requiring output from program
16	Log_{10} number of ADP centers in system
17	Characterize complexity of overall system
21	Log_{10} number of instructions written for this program (POL)
30	Log_{10} number of subprograms
31	Log_{10} number of words in data base
32	Log_{10} number of classes of items in data base
33	Log_{10} number of words in tables and constants not in data base
34	Log_{10} number of input message types
35	Log_{10} number of output message types
36	Characterize complexity of program design
37	Percent clerical instructions
38	Percent math instructions
39	Percent I/O instructions
40	Percent logical control instructions
41	Percent self-checking-fix instructions
42	Percent information storage and retrieval
43	Percent data acquisition and display
45	Percent decision making
46	Percent transformation-reformatting data
47	Percent generation to produce desired output
50	Insufficient memory capacity
51	Insufficient I/O capacity
52	Stringent timing requirements

ID. No.	Title
64	First programming effort on computer
65	Log_{10} average turnaround time experienced by programmers
66	Log_{10} number of ADP components developed with program
87	Log_{10} average turnover rate
99	Number of agencies whose concurrence was required on operational design
100	Estimated customer experience
101	Was computer operated by agency other than program developer
102	Open/closed shop
103	Time sharing
104	Program developed away from operational location
105	Computer at operation site different than at development site
106	Program developed at more than one location
116	Log_{10} miles traveled
117	Log_{10} travel rate
118	Log_{10} pages documentation
122	Log_{10} staffing stability index
125	Log_{10} throughput index
126	Log_{10} data retrieval index
127	Log_{10} data turnover index
130	Percent new instructions (machine)
132	Percent reused instructions (machine)
134	Management index
138	Log_{10} senior man months
141	Log_{10} average senior programmer experience
142	Log_{10} average junior programmer experience
143	Percent programmer participating in design
144	Percent programmers for duration of project
1	Log_{10} total man months
9	Log_{10} total computer hours
20	Log_{10} new instructions written (machine)

ID. No.	Title
54	Log_{10} number of instructions in support programs
109	Percent error rate (machine)
111	Log_{10} production rate (machine)
115	Percent operational discards
119	Log_{10} documentation rate (machine)
121	Log_{10} months elapsed
123	Log_{10} computer usage rate (machine)
128	Percent senior programmers
140	Log_{10} average programmer experience
113	Log_{10} percent utility instructions (machine)

APPENDIX VI

A LIST OF MAJOR INDEPENDENT VARIABLES

The 36 predictor variables listed in this Appendix were obtained from the 69 dependent and independent variables listed in Appendix V by eliminating the primary dependent variables, and other variables that also represented resource-costs, e.g., Number of Trips.

ID. No.	Title
12	Innovation in system
17	Characterize complexity of overall system
30	Log_{10} number of subprograms
31	Log_{10} number of words in data base
32	Log_{10} number of classes of items in data base
33	Log_{10} number of words in tables and constants not in data base
34	Log_{10} number of input message types
35	Log_{10} number of output message types
36	Characterize complexity of program design
37	Percent clerical instructions
38	Percent math instructions
39	Percent I/O instructions
40	Percent logical control instructions
41	Percent self-checking-fix instructions
42	Percent information storage and retrieval
47	Percent generation to produce desired output
50	Insufficient memory capacity
51	Insufficient I/O capacity
52	Stringent timing requirements
64	First programming effort on computer
65	Log_{10} average turnaround time experienced by programmers
66	Log_{10} number of ADP components developed with program
100	Estimated customer experience
101	Was computer operated by agency other than developer
102	Open/closed shop
103	Time sharing
104	Program developed away from operational location
105	Computer at operational site different than at development site
106	Program developed at more than one location
22	Log_{10} number of reused instructions (machine)
134	Management index
143	Percent programmers participating in design

ID. No.	Title
109	Percent error rate (machine)
111	Log_{10} production rate (machine)
115	Percent operational discards
128	Percent senior programmers

(Page 164 Blank)

APPENDIX VII

FACTOR LOADINGS OF SELECTED VARIABLES AND

DERIVATION OF THE COSTLINESS FACTOR

This Appendix summarizes the results of the factor analysis of 69 dependent and independent variables, which was used to obtain the Costliness Factor. The variables considered in the factor analysis are given, as well as the loadings for the first factor. Also, the statistical relationships resulting from the regression of Costliness Factor against its components, to obtain proper weighting, are shown.

APPENDIX VII (Cont'd)

ID. No.	Title	Factor Loadings
10	Log_{10} number of trips	73
11	Log_{10} average distance/trip	51
12	Innovation in system	32
13	Participation by programming organization	20
14	How well were plans known and documented	-5
15	Log_{10} number of users supplying input	23
16	Log_{10} number of ADP centers	-20
17	System complexity	54
19	Log_{10} new instructions written (POL)	-2
30	Log_{10} number of subprograms	64
31	Log_{10} number of words in data base	47
32	Log_{10} number of classes in data base	32
33	Log_{10} number of words not in data base	40
34	Log_{10} Number of input message types	41
35	Log_{10} number of output message types	38
36	Complexity of program design	59
37	Percent clerical instructions	-35
38	Percent math instructions	21
39	Percent I/O instructions	4
40	Percent logical control	17
41	Percent self-check-fix	4
42	Percent information, storage and retrieval	-19
43	Percent data acquisition and display	15
45	Percent decision making	2
46	Percent transform-reformatting	-22
47	Percent generation	37
50	Insufficient memory capacity	36
51	Insufficient I/O capacity	28
52	Stringent timing requirements	31
64	First programming effort on computer	27

APPENDIX VII (Cont'd)

ID. No.	Title	Factor Loadings
65	Log_{10} average turnaround time	13
66	Log_{10} number ADP components developed with program	33
87	Log_{10} average turnover rate	-40
99	Number of agencies concurring in design	18
100	Estimate customer experience	-17
101	Was computer operated by agency other than developer	-1
102	Open/close shop	4
103	Time sharing	-23
104	Program developed away from operational location	17
105	Operational computer different than development computer	5
106	Program developed at more than one location	26
116	Log_{10} miles traveled	67
117	Log_{10} travel rate	3
118	Log_{10} pages documentation	80
122	Log_{10} staffing stability index	26
125	Log_{10} throughput index	79
126	Log_{10} data retrieval index	-9
127	Log_{10} data turnover index	29
130	Percent new instructions (machine)	17
132	Percent revised instructions (machine)	-17
134	Management Index	38
138	Log_{10} senior man months	81
141	Log_{10} average senior programmer experience	-34
142	Log_{10} average junior programmer experience	-68
143	Percent programmers participating in design	-46
144	Percent programmers for duration of project	-28
1	Log_{10} total man months	91
9	Log_{10} total computer hours	88
20	Log_{10} new instructions (machine)	88

APPENDIX VII (Cont'd)

ID. No.	Title	Factor Loadings
54	Log_{10} number support instructions	39
109	Percent error rate (machine	6
111	Log_{10} production rate (machine)	13
115	Percent operational discards	23
119	Log_{10} documentation rate (machine)	-19
121	Log_{10} months elapsed	77
123	Log_{10} computer usage rate (machine	-5
128	Percent senior programmers	-14
141	Log_{10} average programmer experience	-68
145	Log_{10} percent utility instructions (machine)	-40

APPENDIX VII (Cont'd)

CORRELATION AND REGRESSION ANALYSIS:

COSTLINESS FACTOR (154), COSTLINESS FACTOR COMPONENTS

		Log_{10} No. of Trips	Log_{10} Pgs. Document.	Log_{10} Total Man Months	Log_{10} Total Comp. Hrs.	Log_{10} New Instr.
Intercorrelations of Unweighted Components						
10	Log_{10} number of trips	1.00	.51	.62	.61	.55
118	Log_{10} pages documented	.51	1.00	.81	.72	.74
1	Log_{10} total man months	.62	.81	1.00	.86	.87
9	Log_{10} total computer hours	.61	.72	.86	1.00	.86
20	Log_{10} new instructions (machine)	.55	.74	.87	.86	1.00
Table of Validity and Correlation Coefficient						
	Validity of adjusted components with costliness factor	.73	.80	.91	.88	.88
	Adjusted "b" weights	6.23	3.34	8.44	4.48	6.34

Adjusted Multiple Correlation Coefficient

.96

Adjusted Standard Error of Estimate

5.74

Adjusted Prediction Equation

$$Y_{154} = 6.23X_{10} + 3.34X_{118} + 8.44X_{1} + 4.48X_{9} + 6.34X_{20} - 58.63$$

(Page 170 Blank)

APPENDIX VIII

CORRELATION AND REGRESSION ANALYSIS: THE DEVELOPMENT OF EQUATIONS FOR ESTIMATING THE MAJOR DEPENDENT VARIABLES USING SELECTED PREDICTORS

This Appendix provides the details of four estimating equations that use six to eight individual variables as predictors, instead of task indices. The equations are comparable, in this sense, to those of the previous analysis of this series (TM-1447/001/00), expect that Number of New Instructions is not used as a predictor.

APPENDIX VIII (Cont'd)

VARIABLE 1, LOG_{10} TOTAL MAN MONTHS AGAINST SELECTED PREDICTORS

		Innov. in System	Log_{10} Num. Subprog.	% Clerical	% Gen.	First Prog. Effort on Comp.	Est. Cus. Exp.	Prog. Dev. at More Than One Loc.	% Prog. Part. in Design
Table of Intercorrelations									
12	Innovation in System	1.00	.00	.13	.17	.22	- .16	- .11	.06
30	Log_{10} Number of Subprograms	.00	1.00	- .28	.14	- .02	.27	.30	- .39
37	% Clerical Instructions	.13	- .28	1.00	- .14	.06	- .11	- .17	- .03
47	% Generation	.17	.14	- .14	1.00	.00	- .16	- .12	- .08
64	First Programming Effort on Computer	.22	- .02	.06	.00	1.00	- .30	- .04	- .16
100	Estimated Customer Experience	- .16	.27	- .11	- .16	- .30	1.00	.05	.04
106	Program Developed at More Than One Location	- .11	.30	- .17	- .12	- .04	.05	1.00	- .17
143	% Programmers Participating in Design	.06	- .39	- .03	- .08	- .16	.04	- .17	1.00
Table of Validity and Regression Coefficients									
Validity with Log_{10} Total Man Months		.23	.46	- .25	.36	.30	- .29	.27	- .45
Beta Weight		.17	.30	- .18	.23	.16	- .28	.16	- .27
"b" Weight		.26	.41	- .53	1.06	.28	- .36	.25	- .46

Multiple Correlation Coefficient

.75

Standard Error of Estimate

.51

Prediction Equation

$$Log_{10} Y_1 = .26X_{12} + .41X_{30} - .53X_{37} + 1.06X_{47} + .28X_{64} - .36X_{100} + .25X_{106} - .46X_{143} + 1.99$$

VARIABLE 20, LOG_{10} NEW INSTRUCTIONS AGAINST SELECTED PREDICTORS

		Innov. in System	Log_{10} Num. Subprog.	% Gen.	First Prog. Effort on Comp.	Est. Cus. Exp.	Comp. at Oper. Site Diff. From Dev.	Prog. Dev. at More Than One Loc.
Table of Intercorrelations								
12	Innovation in System	1.00	.00	.17	.22	- .16	- .11	.06
30	Log_{10} Number of Subprograms	.00	1.00	.14	- .02	.27	.30	- .39
47	% Generation	.17	.14	1.00	.00	- .16	- .12	- .08
64	First Programming Effort on Computer	.22	- .02	.00	1.00	- .30	- .04	- .16
100	Estimated Customer Experience	- .16	.27	- .16	- .30	1.00	.05	.04
105	Computer at Operation Site Different from Development Site	- .11	.30	- .12	- .04	.05	1.00	- .17
106	Program Developed at More Than One Location	.06	- .39	- .08	- .16	.04	- .17	1.00
Table of Validity and Regression Coefficients								
Validity with Log_{10} New Instructions		.34	.44	.35	.32	- .19	.24	- .36
Beta Weight		.26	.32	.25	.21	- .14	.19	- .15
"b" Weight		.38	.42	1.17	.35	- .17	.28	- .25

Multiple Correlation Coefficient

.71

Standard Error of Estimate

.54

Prediction Equation

$$Log_{10} Y_{20} = .38X_{12} + .42X_{30} + 1.17X_{47} + .35X_{64} - .17X_{100} + .28X_{105} - .25X_{106} + 3.45$$

APPENDIX VIII (Cont'd)

VARIABLE 121, LOG_{10} MONTHS ELAPSED AGAINST SELECTED PREDICTORS

	Log_{10} Num. Subprog.	Log_{10} Num. of Words in Data Base	Log_{10} Num. of Output Messages	Stringent Timing Req.	First Prog. Effort on Comp.	Log_{10} Ave. Turnaround Time
Table of Intercorrelations						
30 Log_{10} Number of Subprograms	1.00	.30	.39	.42	- .02	- .19
31 Log_{10} Number of Words in Data Base	.30	1.00	.21	- .11	.12	.08
35 Log_{10} Number of Output Messages	.39	.21	1.00	.27	- .18	- .16
52 Stringent Timing Requirements	.42	- .11	.27	1.00	- .13	- .15
64 First Programming Effort on Computer	- .02	.12	- .18	- .13	1.00	.01
65 Log_{10} Average Turnaround Time	- .19	.08	- .16	- .15	.01	1.00
Table of Validity and Regression Coefficients						
Validity with Log_{10} Months Elapsed	.55	.34	.12	.27	.27	.15
Beta Weight	.50	.17	- .08	.18	.27	.24
"b" Weight	.30	.04	- .04	.17	.20	.11

Multiple Correlation Coefficient

.69

Standard Error of Estimate

.24

Prediction Equation

$$Log_{10} Y_{121} = .30X_{30} + .04X_{31} - .04X_{35} + .17X_{52} + .20X_{64} + .11X_{65} + .33$$

VARIABLE 9, LOG_{10} TOTAL COMPUTER HOURS AGAINST SELECTED PREDICTORS

		Innov. in System	Log_{10} Num. Subprog.	% Gen.	Stringent Timing Req.	First Prog. Effort on Comp.	% Prog. Part. in Design
Table of Intercorrelations							
12	Innovation in System	1.00	.00	.17	.09	.22	.06
30	Log_{10} Number of Subprograms	.00	1.00	.14	.42	- .02	- .39
47	% Generation	.17	.14	1.00	- .08	.00	- .08
52	Stringent Timing Requirement	.09	.42	- .08	1.00	- .13	- .15
64	First Programming Effort on Computer	.22	- .02	.00	- .13	1.00	- .16
143	% Programmers Participating in Design	.06	- .39	- .08	- .15	- .16	1.00
Table of Validity and Regression Coefficients							
	Validity with Log_{10} Total Computer Hours	.35	.57	.35	.27	.35	- .51
	Beta Weight	.26	.39	.24	.10	.26	- .30
	"b" Weight	.42	.57	1.24	.16	.49	- .55

Multiple Correlation Coefficient

.81

Standard Error of Estimate

.50

Prediction Equation

$$Log_{10}\ Y_9 = .42X_{12} + .57X_{30} + 1.24X_{47} + .16X_{52} + .49X_{64} - .55X_{143} + 1.35$$

(Page 176 Blank)

APPENDIX IX

CORRELATION AND REGRESSION ANALYSIS:

THE DERIVATION OF WEIGHTS FOR THE COMPONENTS OF THE UNIQUENESS INDEX

This Appendix illustrates the procedures that were used to obtain weights for the components of each of the task indices. The Uniqueness Index, variable 149, is used as an example.

The five major dependent variables were first regressed against the components of the Uniqueness Index. The details are shown in this Appendix for each case. Next, the regression coefficients were averaged and rescaled to obtain final weights. This process is also detailed.

APPENDIX IX (Cont'd)

VARIABLE 1, LOG_{10} TOTAL MAN MONTHS

AGAINST COMPONENTS OF UNIQUENESS INDEX

		Innovation	Stringent Timing	First Effort on Computer	Prog. Dev. at More Than 1 Loc.
Intercorrelation of Unweighted Uniqueness Components					
12	Innovation in System	1.00	.09	.22	- .11
52	Stringent Timing Requirements	.09	1.00	- .13	.18
64	First Programming Effort on Computer	.22	- .13	1.00	- .04
106	Program Developed at More Than One Location	- .11	.18	- .04	1.00
Table of Validity and Regression Coefficients					
Validity of Unweighted Uniqueness Components with Log_{10} Total Man Months		.23	.21	.30	.27
"b" Weights		.27	.28	.51	.41

Multiple Correlation Coefficient with Adjusted "b" Weights

.49

Standard Error of Estimate with Adjusted "b" Weights[20]

.68

[20] The computation of the adjusted "b" weights is shown in the last page of this Appendix.

APPENDIX IX (Cont'd)

VARIABLE 9, LOG_{10} TOTAL COMPUTER HOURS

AGAINST COMPONENTS OF UNIQUENESS INDEX

	Innovation	Stringent Timing	First Effort on Computer	Prog. Dev. at More Than 1 Loc.
Intercorrelation of Unweighted Uniqueness Components				
12 Innovation in System	1.00	.09	.22	- .11
54 Stringent Timing Requirements	.09	1.00	- .13	.18
64 First Programming Effort on Computer	.22	- .13	1.00	- .04
106 Program Developed at More Than One Location	- .11	.18	- .04	1.00
Table of Validity and Regression Coefficients				
Validity of Unweighted Uniqueness Components with Log_{10} Computer Hours	.35	.27	.35	.20
"b" Weights	.45	.42	.60	.34

Multiple Correlation Coefficient with Adjusted "b" Weights

.56

Standard Error of Estimate with Adjusted "b" Weight[21]

.70

---

[21] The computation of the adjusted "b" weights is shown in the last page of this Appendix.

APPENDIX IX (Cont'd)

VARIABLE 20, LOG_{10} NEW MACHINE INSTRUCTIONS

AGAINST COMPONENTS OF UNIQUENESS

		Innovation	Stringent Timing	First Effort on Computer	Prog. Dev. at More Than 1 Loc.
Intercorrelation of Unweighted Uniqueness Components					
12	Innovation in System	1.00	.09	.22	- .11
54	Stringent Timing Requirements	.09	1.00	- .13	.18
64	First	.22	- .13	1.00	- .04
106	Program Developed at More Than One Location	- .11	.18	- .04	1.00
Table of Validity and Regression Coefficients					
Validity of Unweighted Uniqueness Components with Log_{10} New Machine Instructions		.34	.16	.32	.24
"b" Weights		.43	.19	.46	.39

Multiple Correlation Coefficients with Adjusted "b" Weight

.49

Standard Error of Estimate with Adjusted "b" Weight[22]

.64

[22] The computation of the adjusted "b" weights is shown in the last page of this Appendix.

APPENDIX IX (Cont'd)

VARIABLE 121, LOG_{10} MONTHS ELAPSED

AGAINST COMPONENTS OF UNIQUENESS INDEX

		Innovation	Stringent Timing	First Effort on Computer	Prog. Dev. at More Than 1 Loc.
Intercorrelation of Unweighted Uniqueness Components Variable Number					
12	Innovation in System	1.00	.09	.22	- .11
54	Stringent Timing Requirements	.09	1.00	- .13	.18
64	First Programming Effort on Computer	.22	- .13	1.00	- .04
106	Program Developed at More Than One Location	- .11	.18	- .04	1.00
Table of Validity and Regression Coefficients					
Validity of Unweighted Uniqueness Components with Log_{10} Total Man Months		.19	.27	.27	.29
"b" Weights		.08	.17	.22	.18

Multiple Correlation Coefficients with Adjusted "b" Weights

.50

Standard Error of Estimate with Adjusted "b" Weight[23]

.30

[23] The computation of the adjusted "b" weights is shown in the last page of this Appendix.

APPENDIX IX (Cont'd)

VARIABLE 154, COSTLINESS FACTOR

AGAINST COMPONENTS OF UNIQUENESS INDEX

		Innovation	Stringent Timing	First Effort on Computer	Prog. Dev. at More Than 1 Loc.
Intercorrelation of Unweighted Uniqueness Components					
12	Innovation in System	1.00	.09	.22	- .11
52	Stringent Timing Requirement	.09	1.00	- .13	.18
64	First Programming Effort on Computer	.22	- .13	1.00	- .04
106	Program Developed at More Than One Location	- .11	- .18	- .04	1.00
Table of Validity and Regression Coefficients					
Validity of Unweighted Uniqueness Components with Costliness Factor		.29	.24	.30	.27
"b" Weights		.45	.40	.62	.52

Multiple Correlation Coefficient with Adjusted "b" Weights

.52

Standard Error of Estimate with Adjusted "b" Weights[24]

.84

---

[24] The computation of the adjusted "b" weights is shown in the last page of this Appendix.

APPENDIX IX (Cont'd)

COMPUTATION OF ADJUSTED WEIGHTS

FOR COMPONENTS OF THE UNIQUENESS INDEX

After the five major dependent variables were regressed against the components of the Uniqueness Index, the resulting coefficients, or "b" weights, for each component were averaged and divided by the smallest value to obtain the final weights.

REGRESSION COEFFICIENTS

Major Cost Variables	Innovation	Stringent Timing	First Programming Effort on Computer	Program Developed at More Than 1 Location
Log_{10} Man Months	.27	.28	.51	.41
Log_{10} Computer Hours	.45	.42	.60	.34
Log_{10} New Machine Instructions	.43	.19	.46	.39
Log_{10} Months Elapsed	.08	.17	.22	.18
Costliness Factor	.45	.40	.62	.52
Sums of "b" Weights	1.68	1.46	2.41	1.84
Component Weights	1.2	1.00	1.7	1.3

(Page 184 Blank)

APPENDIX X

CORRELATION AND REGRESSION ANALYSIS:

THE DEVELOPMENT OF ESTIMATION EQUATIONS

FOR THE MAJOR DEPENDENT VARIABLES

USING TASK INDICES AS PREDICTORS

This Appendix provides the statistical details that underlie the estimation equations developed for each of the major dependent variables, i.e., Log_{10} Total Man Months, Log_{10} Total Computer Hours, Log_{10} New Machine Instructions, Log_{10} Months Elapsed, and the Costliness Factor, with the indices--Uniqueness, Job Difficulty, Development Environment and Type of Job--used as a predictor variables.

The reader should note that the Standard Errors of Estimate are in logarithmic form.

APPENDIX X (Cont'd)

VARIABLE 1, LOG_{10} TOTAL MAN MONTHS

AGAINST TASK INDICES

	Uniqueness	Job Difficulty	Development Environment	Type of Job
Table of Intercorrelations				
149 Uniqueness Index	1.00	.28	+ .25	.15
150 Job Difficulty Index	.28	1.00	+ .28	.24
151 Development Environment Index	- .25	- .28	1.00	- .14
152 Type of Job Index	.15	.24	+ .14	1.00
Table of Validity and Regression Coefficients				
Validity Coefficients with Log_{10} Man Months	.49	.48	+ .52	.44
Beta Weight	.30	.23	+ .33	.29
"b" Weight	.17	.06	+ .16	.27
Adjusted "b" Weight	.23	.08	+ .21	.36

Multiple Correlation Coefficient

.75

Standard Error of Estimate

.52

Adjusted Standard Error of Estimate

.54

Prediction Equation

$$Log_{10} Y_1 = .17X_{.49} + .06X_{150} + .16X_{151} + .27X_{152} - 1.54$$

Adjusted Prediction Equation

$$Log_{10} Y_1 = .23X_{149} + .08X_{150} + .21X_{151} + .36X_{152} + 1.55$$

APPENDIX X (Cont'd)

VARIABLE 9, LOG_{10} TOTAL COMPUTER HOURS

AGAINST TASK INDICES

	Uniqueness	Job Difficulty	Development Environment	Type of Job
Table of Intercorrelations				
149 Uniqueness Index	1.00	.28	+ .25	.15
150 Job Difficulty Index	.28	1.00	+ .28	.24
151 Development Environment Index	- .25	- .28	1.00	- .14
152 Type of Job Index	.15	.24	+ .14	1.00
Table of Validity and Regression Coefficients				
Validity Coefficients with Log_{10} Total Computer Hours	.56	.58	+ .51	.37
Beta Weight	.36	.35	+ .29	.19
"b" Weight	.22	.10	+ .15	.19
Adjusted "b" Weight	.28	.12	+ .19	.25

Multiple Correlation Coefficient

.79

Standard Error of Estimate

.52

Adjusted Standard Error of Estimate

.53

Prediction Equation

$$Log_{10} Y_9 = .22X_{149} + .10X_{150} + .15X_{151} + .19X_{152} + 1.78$$

Adjusted Prediction Equation

$$Log_{10} Y_9 = .28X_{199} + .12X_{150} + .19X_{151} + .25X_{152} + 1.69$$

APPENDIX X (Cont'd)

VARIABLE 20, LOG_{10} NEW MACHINE INSTRUCTIONS AGAINST TASK INDICES

	Uniqueness	Job Difficulty	Development Environment	Type of Job
Table of Intercorrelations				
149 Uniqueness Index	1.00	.28	+ .25	.15
150 Job Difficulty Index	.28	1.00	+ .28	.24
151 Development Environment Index	- .25	- .28	1.00	- .14
152 Type of Job Index	.15	.24	+ .14	1.00
Table of Validity and Regression Coefficients				
Validity Coefficients with Log_{10} New Instructions	.52	.47	+ .40	.39
Beta Weight	.34	.26	+ .25	.26
"b" Weight	.19	.06	+ .11	.23
Adjusted "b" Weights	.29	.09	+ .13	.32

Multiple Correlation Coefficient

.70

Standard Error of Estimate

.54

Adjusted Standard Error of Estimate

.57

Prediction Equation

$$Log_{10} Y_{20} = .19X_{149} + .06X_{150} + .11X_{151} + .23X_{152} + 3.46$$

Adjusted Prediction Equation

$$Log_{10} Y_{20} = .29X_{149} + .09X_{150} + .13X_{151} + .32X_{152} + 3.31$$

APPENDIX X (Cont'd)

VARIABLE 121, LOG_{10} MONTHS ELAPSED

AGAINST TASK INDICES

	Uniqueness	Job Difficulty	Development Environment	Type of Job
Table of Intercorrelations				
149 Uniqueness Index	1.00	.28	+ .25	.15
150 Job Difficulty Index	.28	1.00	+ .28	.24
151 Development Environment Index	- .25	- .28	1.00	- .14
152 Type of Job Index	.15	.24	+ .14	1.00
Table of Validity and Regression Coefficients				
Validity Coefficients with Log_{10} Months Elapsed	.48	.55	+ .38	.36
Beta Weight	.31	.36	+ .18	.21
"b" Weight	.08	.04	+ .04	.08
Adjusted "b" Weight	.11	.06	+ .05	.12

Multiple Correlation Coefficient

.70

Standard Error of Estimate

.25

Adjusted Standard Error of Estimate

.26

Prediction Equation

$$Log_{10} Y_{121} = .08X_{149} + .04X_{150} + .04X_{151} + .08X_{152} + .70$$

Adjusted Prediction Equation

$$Log_{10} Y_{121} = .11X_{149} + .06X_{150} + .05X_{151} + .12X_{152} + .12$$

APPENDIX X (Cont'd)

VARIABLE 154, COSTLINESS FACTOR

AGAINST TASK INDICES

	Uniqueness	Job Difficulty	Development Environment	Type of Job
Table of Intercorrelations				
149 Uniqueness Index	1.00	.28	- .25	.15
150 Job Difficulty Index	.28	1.00	- .28	.24
151 Development Environment Index	- .25	- .28	1.00	- .14
152 Type of Job Index	.15	.24	- .14	1.00
Table of Validity and Regression Coefficients				
Validity Coefficients with Costliness Factor	.52	.59	- .53	.45
Beta Weight	.30	.35	- .32	.28
"b" Weight	5.83	2.90	-3.81	8.39
Adjusted "b" Weight	18.93	3.58	-5.89	10.37

Multiple Correlation Coefficient

.82

Standard Error of Estimate

12.0

Adjusted Standard Error of Estimate

12.7

Prediction Equation

$$Y_{154} = 5.83X_{149} + 2.90X_{150} + 3.81X_{151} + 8.39X_{152} + 88.47$$

Adjusted Prediction Equation

$$Y_{154} = 18.93X_{149} + 3.58X_{150} + 5.89X_{151} + 10.37X_{152} + 66.96$$

APPENDIX XI

THE COMPUTATIONAL PROCEDURES TO CORRECT FOR CURVILINEARITY IN THE STANINE BANDS

The Stanine bands that are shown in Figures 14, 15, 16, 17, and 18, Section XVII, are straight-line approximations based on constant multiples of the Standard Error of Estimate (σ_E), e.g., the limits of the "5" band are $\pm$ 0.25σ_E about the prediction line.

In fact, the Standard Error of Estimate is not constant but expands as the predictor indices differ from their respective means. The result is called the Standard Error of Prediction (or, sometimes, the Standard Error of Forecast). Since the corrections are unique for every combination of predictor values, the calculations are rather complex(14).

The reader should carefully note, however, that both the dependent variables and the Standard Errors in the five estimation equations are in logarithmic form. This means that, when the Stanine band limits are computed (by adding and subtracting an appropriate multiple of the Standard Error from the equation value), it is necessary to add and subtract these values retained in their logarithmic form before converting to raw values.

The combination of these procedures produces corrections that are best illustrated with a pair of examples in the following table. These have been worked out for two cases of the dependent variable Total Man Months, one at 33 man months, and the other at 540 man months. In the first example, all the predictor indices are near their respective means.

Several aspects of the computed corrections are worthy of note:

a. The percentage corrections are quite small when the predictor indices are near their respective means, but enlarge as the predictors deviate from their means. The corrections are added to the computed estimate above the prediction line and subtracted below it.

b. The use of logarithmically transformed variables also results in an increased percentage correction the farther a particular band is from the prediction line, e.g., a larger correction at the "7" band than at the "5" band. These errors are also asymmetrical in raw values, although the percentage corrections are symmetrical. (see the last two columns in the charts for each example.)

Although these two examples do not demonstrate the point, the percentage corrections (see the 540-man-month case in the chart) level off i.e., reach an asymptote for larger values, while the raw errors continue to increase proportionately (The same is true for cases below 33 man months, although we have not worked out an example).

To allow the reader to determine the corrections for an individual estimation and Stanine band, a work sheet has been provided. In addition to the computed value of each index, the Standard Error of Estimate is required for each equation, and has been abstracted from Appendix X.

Dependent Variable	Standard Error of Estimate (in Log_{10})
Total Man Months	0.54
Total Computer Hours	0.53
New Machine Instructions	0.57
Months Elapsed	0.26
Costliness Factor	0.59

To obtain the corrected limits of the Stanine bands for an individual case, the Standard Error of Prediction should be multiplied by the following values:

Stanine Band	Multiple of the Standard Error of Prediction
9 (upper limit)	Open-Ended
8 (upper limit	+1.75
7 (upper limit)	+1.25
6 (upper limit	+0.75
5 (upper limit)	+0.25
5 (lower limit)	-0.25
4 (lower limit)	-0.75
3 (lower limit)	-1.25
2 (lower limit)	-1.75
1 (lower limit)	Open-Ended

APPENDIX XI (Cont'd)

A COMPARISON OF STANINE BAND LIMITS BASED ON THE STANDARD ERROR OF ESTIMATE AND THE STANDARD ERROR OF PREDICTION

Program A--Estimated 33 Man Months

Stanine Bands	Estimate Based[24] on Straight-line Approximation (Man Months)	Estimate Adjusted[24] for Curvilinearity (Man Months)	Correction as a Percent of Straight-line Approximation
9 (upper limit)	Open Ended	Open Ended	-
8 (upper limit)	266.90	270.80	1.5
7 (upper limit)	246.80	148.30	1.0
6 (upper limit)	80.80	81.40	0.7
5 (upper limit)	44.40	44.50	0.2
5 (lower limit)	24.50	24.40	-0.2
4 (lower limit)	13.50	13.40	-0.7
3 (lower limit)	7.40	7.32	-1.0
2 (lower limit)	4.07	4.01	-1.5
1 (lower limit)	Open Ended	Open Ended	-

Program B--Estimated 540 Man Months

Stanine Bands	Estimate Based on Straight-line Approximation (Man Months)	Estimate Adjusted for Curvilinearity (Man Months)	Correction as a Percent of Straight-line Approximation
9 (upper limit)	Open Ended	Open Ended	-
8 (upper limit)	4371	4933	13
7 (upper limit)	2404	2622	9
6 (upper limit)	1324	1393	5
5 (upper limit)	728	741	2
5 (lower limit)	401	394	-2
4 (lower limit)	221	209	-5
3 (lower limit)	121	111	-9
2 (lower limit)	67	60	-13
1 (lower limit)	Open Ended	Open Ended	-

[24]Computations were carried out with greater precision near the mean to preserve the accuracy of the comparisons.

APPENDIX XI (Cont'd)

A WORKSHEET FOR COMPUTING THE STANDARD ERROR OF PREDICTION FROM THE STANDARD ERROR OF ESTIMATE WHEN USING TASK INDICES AS PREDICTORS

Indices	Values of Predictor Indices (X_i)	Means of Predictor Indices ($\bar{X}_i$)	Deviation of Predictors from their Means $X_j = (X_i - \bar{X}_i)$
Uniqueness Index		2.09189	
Job Difficulty Index		5.98684	
Development Environment Index		-4.74220	
Job Type Index		.04493	

x_j	x_k	Coefficients of Multiplication d_{jk}	Result of Multiplying $(x_j)(x_k)(d_{jk})$
$(x_1 - \bar{x}_1) =$	$(x_1 - \bar{x}_1) =$	.00887	
$(x_2 - \bar{x}_2) =$	$(x_2 - \bar{x}_2) =$	.00180	
$(x_3 - \bar{x}_3) =$	$(x_3 - \bar{x}_3) =$	.00578	
$(x_4 - \bar{x}_4) =$	$(x_4 - \bar{x}_4) =$	.02196	
$(x_1 - \bar{x}_1) =$	$(x_2 - \bar{x}_2) =$	-.00165	
$(x_1 - \bar{x}_1) =$	$(x_3 - \bar{x}_3) =$	-.00250	
$(x_1 - \bar{x}_1) =$	$(x_4 - \bar{x}_4) =$	-.00209	
$(x_2 - \bar{x}_2) =$	$(x_3 - \bar{x}_3) =$	-.00138	
$(x_2 - \bar{x}_2) =$	$(x_4 - \bar{x}_4) =$	-.00236	
$(x_3 - \bar{x}_3) =$	$(x_4 - \bar{x}_4) =$	-.00150	

Sum of $(x_j)(x_k)(d_{jk})$ =

Add Constant = 1.01351

T =

$\sqrt{T}$ =

Enter Standard Error of Estimate (Log_{10}) =

Multiply by $\sqrt{T}$ to obtain Standard Error of Prediction (Log_{10}) =

XII

THE SENSITIVITY OF THREE DEPENDENT VARIABLES TO CHANGES IN THE TASK INDICES: A SUMMARY

This Appendix summarizes the sensitivity of the dependent variables--Total Man Months, Months Elapsed, and Total Computer Hours--to changes in the predictor indices. Briefly, the computational procedure involves a determination of the equation estimate for the dependent variable with the four predictor indices equal to their respective means, and, then, varying each index by +10 percent, +1 standard deviation, -10 percent, and -1 standard deviation, while holding the remaining 3 predictor indices constant.

APPENDIX XII (Cont'd)

SUMMARY OF SENSITIVITY RELATIONSHIPS

COST FACTOR	INDEX	PERCENT CHANGE IN COST FACTOR FOR INDICATED CHANGE FROM THE MEAN OF THE INDEX			
		-1 Standard Deviation	-10%	+10%	+1 Standard Deviation
Total Man Months (Variable 1)	Uniqueness Index (149)	-40	-8	9	68
	Job Difficulty (150)	-34	-8	9	51
	Development Environment (151)	+83	+19	-16	-45
	Type of Job (152)	-40	negligible	negligible	66
Months Elapsed (Variable 121)	Uniqueness	-22	-4	4	28
	Job Difficulty	-24	-5	6	32
	Development Environment	+16	5	-4	-14
	Type of Job	-13	negligible	negligible	16
Total Computer Hours (Variable 9)	Uniqueness	-49	-9	11	96
	Job Difficulty	-50	-13	15	100
	Development Environment	+76	18	-15	-43
	Type of Job	-30	negligible	negligible	43

REFERENCES

1. In a 1963 article ("A Profile of the Programmer," Industrial Relations News, August, 1963), Deutch and Shea estimated the 1970 requirement for programmers to be in the neighborhood of 200,000. About a year later, Brandon ("The Computer Personnel Revolution," Computers and Automation, August, 1964) projected a need for about 145,000 programmers and 90,000 system analysts in 1970, based on the computer installations anticipated through that time period. Doubling an average salary of $10,000 (which is estimated for 1970 based on an extensive annual SDC National Salary Survey for digital computing personnel) to obtain gross costs, and assuming a demand for 200,000 programmers and system analysts, this yields annual expenditure of $4 billion for computer programming alone by 1970.

Such an estimate is comparable to those contained in the 1963 "Survey and Study of the Computer Field" by the Investment Bankers Association of America (contained in Use of Electronic Data Processing Equipment in the Federal Government, and the more recent Review of Problems Relating to Management and Administration of Electronic Data Processing Systems in the Federal Government. (The Comptroller General of the United States, August 1964.) Assuming the rule of thumb that investments in program development usually equal or exceed those in computer hardware, the Investment Bankers project an annual expenditure range of from $4 to $7 billion by 1970, while the Comptroller General's 1964 Report is somewhat lower at $3 billion. In a more recent report (Management of Automatic Data Processing Facilities in the Federal Government), August 1965, the Comptroller General cites annual expenses for hardware, software, and services by the Federal Government as $3 billion annually.

2. Refers to such reports as the following: Hearing Before the Subcommittee on Census and Government Statistics, Committee on Post Office and Civil Service, House of Representatives, Second Session of the Eighty-Seventh and First Session of the Eighty-Eighth Congress, Use of Electronic Data Processing Equipment, Washington: U. S. Government Printing Office, 1962 and 1963; the Comptroller General of the United States, Review of Problems Relating to Management and Administration of Electronic Data Processing Systems in the Federal Government, April 1964; Committee on Government Operations, United States Senate, First Session of the Eighty-Ninth Congress, Report to the President on the Management of Automatic Data Processing in the Federal Government, Washington: U. S. Government Printing Office, 1965; the Comptroller General of the United States, Management of Automatic Data Processing Facilities in the Federal Government, Washington: U. S. Government Printing Office, 1965; Hearings before a Subcommittee of the Committee on Government Operations, House of Representatives, First Session of the Eighty-Ninth Congress, H.R. 4845- A Bill to Provide for the Economic and Efficient Purchase, Lease, Maintenance, Operation, and Utilization of Automatic Data Processing Equipment by Federal Departments and Agencies, Washington: U. S. Government Printing Office, 1965;

Gill, W. "Sound Management and Effective Use of Computers in the Federal Government," Computers and Automation, April 1965, pp. 15-17.

3. IFIP/ICC Vocabulary of Information Processing. To be published by the North-Holland Publishing Company, Amsterdam, in 1965.

4. Farr, L., LaBolle, V., and Willmorth, N. E. Planning Guide for Computer Program Development. System Development Corporation, TM-2314/000/00, 10 May 1965.

5. Nelson, E. A., and Weinwurm, G. F. Research into the Management of Computer Programming: A Survey of Programming Management Practices in Government and Industry. System Development Corporation, TM-2718, to be published in late 1965.

6. Gradwohl, A. J., Lackner, M. R., Rosen, W. A., and Shelton, W. V. Use of Air Force ADPS Experience in Judging Proposals for New Automation. Electronic Systems Division, ESD-TR-65-277, March 1965; Statland, M., et al. An Approach to Computer Installation Performance Effectiveness Evaluation. Electronic Systems Division, ESD-TR-65-276, June 1965.

7. Benson, S. G., Neil, G., and Searle, L. V. Computer Program Acquisition: Data Requirements. System Development Corporation, TM-2547/000/00, 12 July 1965; Benson, S. G., Neil, G., and Searle, L. V. Configuration Management of Computer Programs for Information Systems. System Development Corporation, TM-1918/000/01 (Draft), 12 July 1965.

8. Farr, L., and Zagorski, H. J. Factors That Affect the Cost of Computer Programming: A Quantitative Analysis. System Development Corporation, TM-1447/001/00, 31 August 1964.

9. Nelson, E. A. Research into the Management of Computer Programming: Some Characteristics of Programming Cost Data From Government and Industry. System Development Corporation, TM-2704, 15 November 1965.

10. Ferguson, T. S. "Rules for the Rejection of Outliers." Revue de l'Institute International de Statistique, Vol. 29, 1961, pp. 29-43; Tukey, J. W. "The Future of Data Analysis." Annals of Mathematical Statistics, Vol. 33, March 1962, pp. 1-67.

11. Ezekial, M. Methods of Correlation Analysis. New York, Wiley, 1941.

12. Harman, H. H. Modern Factor Analysis. Chicago, University of Chicago Press, 1960, pp. 154 ff.

13. Harman, H. Op. cit., pp. 301 ff.

14. Anderson, R. L., and Bancroft, T. A. Statistical Theory in Research. New York, McGraw-Hill, 1952.

15. Proceedings of the Second Annual Conference, Computer Personnel Research Group. Silver Springs, Maryland, The Johns Hopkins University, 1964; Computer Personnel Selection and Criterion Development: Description and Classification of Computer Programming and Analyst Jobs. Electronics Personnel Research Group, University of Southern California, December 1963; Perry, D. K. Training and Job Performance Validities of Programmer Trainee Selection Variables. System Development Corporation, TM-2172, 9 December 1964; Perry, D. K., and Cantley, G. Computer Programmer Selection and Training in System Development Corporation. System Development Corporation, TM-2234, 2 February 1965.

16. Hilleglass, J., Statland, M., and Taylor, A. Quantitative Methods for Information Processing Systems Evaluation. Electronic Systems Division, ESD-TR-64-194, January 1964. Also see (6).

17. For some other ideas, see: Tonge, F. M. Some Reflections on a Survey of Research Problems in Computer Software. The RAND Corporation, RM-4467-PR, April 1965; Advanced Programming Developments: A Survey. Electronic Systems Division, ESD-TR-65-171, February 1965.

18. Farr, L., and Zagorski, H. J. Op. cit., pp. 58-86.

19. Farr, L., and Zagorski, H. J. Op. cit., p. 13.

20. The application of the Stanine concept to estimation was developed in connection with a related USAF research study: Sheldon, M. S., and Zagorski, H. J. Project NORM Mission I Research Report. System Development Corporation, TM-2232/001/00, 15 October 1965.

21. For a discussion of the use of sensitivity analysis in the costing of weapons systems and total force structure, see: Novick, System and Total Force Cost Analysis. The RAND Corporation, RM-2695, April 1961, pp. 95 ff; Large, J. P. Concepts and Procedures of Cost Analysis. The RAND Corporation, RM-3589-PR, June 1963, pp. VI-23 ff.

22. Haverty, J. P., and Patrick, R. L. Programming Languages and Standardization in Command and Control. The RAND Corporation, RM-3447-PR, January 1963; Bosak, R., Dobbs, G., Dobrusky, W., Jacobs, E. H., et al. Computer Programming Standards in Command and Control. System Development Corporation, TM-688/000/01, February 1962. Also see: Gray, H. J., and Kapps, C., et al. Interactions of Computer Language and Machine Design. Rome Air Development Center, RADC-TR-64-511, May 1965.

(Page 200 Blank)

BIBLIOGRAPHY

Previous Publications of the Programming Management Project

1. Bleier, R. E. A Description of the Computer Program Implementation Process: A Process Flow. System Development Corporation, TM-1021/003/00, 9 May 1963.

2. Bleier, R. E. Frequency Analysis of Machine Instructions in Computer Program Systems. System Development Corporation, TM-1603, 19 November 1963.

3. Farr, L. A Description of the Computer Program Implementation Process. System Development Corporation, TM-1021/002/00, 25 February 1963.

4. Farr, L. and Nanus, B. Factors That Affect the Cost of Computer Programming. System Development Corporation, TM-1447/000/02, 30 June 1964.

5. Farr, L. and Zagorski, H. J. Factors That Affect the Cost of Computer Programming: A Quantitative Analysis. System Development Corporation, TM-1447/001/00, 31 August 1964.

6. Farr, L. and Zagorski, H. J. A Summary of an Analysis of Computer Programming Cost Factors. System Development Corporation, TM-1447/002/00 25 January 1965.

7. Farr, L., LaBolle, V., Willmorth, N. E. Planning Guide for Computer Program Development. System Development Corporation, TM-2314/000/00, 10 May 1965.

8. Farr, L. and Zagorski, H. J. Quantitative Analysis of Computer Programming Cost Factors: A Progress Report. System Development Corporation, SP-2036, 26 August 1965.

9. Heinz, K., Claussen, N. and LaBolle, V. Management of Computer Programming for Command and Control Systems. System Development Corporation, TM-903/000/02, 8 May 1963.

10. LaBolle, V. Management Aspects of Computer Programming for Command and Control Systems. System Development Corporation, SP-1000/000/02, 5 February 1963.

11. LaBolle, V. Estimation of Computer Programming Costs. System Development Corporation, SP-1747, 14 September 1964.

12. LaBolle, V. Critical Management Points. System Development Corporation, SP-1934, 12 February 1965.

13. Nanus, B. Operations Research Opportunities in Computer Programming Management. System Development Corporation, SP-1768, 14 September 1964.

14. Nelson, E. A. Research Into the Management of Computer Programming: Some Characteristics of Programming Cost Data From Government and Industry. System Development Corporation, TM-2704, 15 November 1965.

15. Nelson, E. A., and Weinwurm, G. F. Research Into the Management of Computer Programming: A Survey of Programming Management Practices in Government and Industry. System Development Corporation, TM-2718, to be published in late 1965.

16. Weinwurm, G. F. Research in the Management of Computer Programming. System Development Corporation, SP-2059, 1 May 1965.

17. Weinwurm, G. F. The Management of Information Processing. System Development Corporation, SP-1992/000/01, 15 July 1965.

General References

1. Anderson, R. L., and Bancroft, T. A. Statistical Theory in Research, New York, McGraw-Hill, 1952.

2. Cattell, R. B. Factor Analysis. New York, Harper, 1952.

3. Chew, V. (Ed.) Experimental Designs in Industry. New York, Wiley, 1958.

4. Davies, O. L. (Ed.) Design and Analysis of Industrial Experiments. Edinburgh, Oliver and Boyd, 1956.

5. Davies, O. L. (Ed.) Statistical Methods in Research and Production. Edinburgh, Oliver and Boyd, 1949.

6. Ezekiel, M. Methods of Correlation Analysis. New York, Wiley, 1941.

7. Farrar, D. E., and Apple, R. E. Some Factors that Affect the Overhaul-Cost of Ships: An Exercise in Statistical Cost Analysis. Naval Research Logistics Quarterly, December 1963.

8. Ferguson, T. S. Rules for Rejection of Outliers, in Revue de l'Institut International de Statistique, Vol. 29, 1961, pp. 29-43.

9. Fruchter, B. Introduction to Factor Analysis. New York, Von Nostrand, 1954.

10. Hald, A. Statistical Theory with Engineering Applications, New York Wiley, 1952.

11. Harman, H. H. Modern Factor Analysis. Chicago, University of Chicago Press, 1960.

12. Johnson, P. O. Statistical Methods in Research. New York, Prentice-Hall, 1950.

13. Lyle, P. Regression Analysis of Production Costs and Factory Operations. Edinburgh, and London, Oliver and Boyd, (third edition revised by H. C. Tippett), 1957.

14. Ostle, B. Statistics in Research. Ames, Iowa State Press, 1963.

15. Richmond, S. B. Statistical Analysis. New York, Ronald Press, 1957.

16. Scheffe, H. The Analysis of Variance. New York, Wiley, 1959.

17. System Reliability Prediction by Function. Volume I, II Development of Prediction Techniques, RADC-TDR-63-300, August 1963.

18. Tukey, J. W. "The Future of Data Analysis." in Annals of Mathematical Statistics, Vol. 33, March 1962, pp. 1-67.

19. Tukey, J. W. Statistical and Quantitative Methodology. Princeton University, Princeton, New Jersey, July 1959.

20. Walker, H. M., and Lev, J. Statistical Inference. New York, Henry Holt, 1953.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

System Development Corporation,
Santa Monica, California

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

RESEARCH INTO THE MANAGEMENT OF COMPUTER PROGRAMMING: A TRANSITIONAL ANALYSIS OF COST PREDICTION TECHNIQUES.

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(Last name, first name, initial)*

Weinwurm, G. F., Zagorski, H. J.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
12 November 1965	203	22

8a. CONTRACT OR GRANT NO. AF 19(628)-5166	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	ESD-TR-65-575
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	SDC TM-2712

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this report is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Director of Computers Electronic Systems Division Bedford, Massachusetts

13. ABSTRACT

The report embodies the latest results of a continuing research effort directed toward the development of management guidelines, standards, and techniques in the field of computer programming. The work is based upon earlier studies at System Development Corporation that included: the definition of variables affecting computer programming costs; the design of a questionnaire as an aid to collecting data on completed jobs; and the exploratory statistical analysis of 27 completed computer programming jobs to develop preliminary cost-estimation relationships. The present report is focused upon the statistical analysis of 74 completed computer programming jobs in terms of their resource-costs and related variables, e.g., man months, computer hours. The primary results developed in this analysis are: indices of job difficulty, job type, development environment, and job uniqueness; a "costliness" factor that permits programming tasks to be ranked in this respect; weighted composites of the indices for estimating the cost of particular programming jobs; and scoring and confidence-band techniques for blending intuitive managerial judgments with the formal cost-estimation procedures. Supplementary findings include indications of the relative sensitivity of job cost to changes in the values for the indices, and preliminary comparisons of resource usage between programs produced in machine-oriented or procedure-oriented languages. Also, recommendations are made for future research, including: the collection of more accurate and current data on programming jobs during the production cycle, and the development of a census of computer programming, to enable the design of precise sampling experiments for subsequent analyses. (author)

DD FORM 1 JAN 64 1473

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Management Computer Programming Costs Statistical Analysis						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.